Možnosti využití technologie XForms a XML+XSLT v PHP web aplikacích

Posibilities of using XForms and XML+XSLT in PHP web applications

Bc. Radek Vala

Diplomová práce
2009

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Radek VALA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Možnosti využití technologie XForms a XML+XSLT v PHP web aplikacích**

Zásady pro vypracování:

1. **Prostudujte technologie XML, XSLT a XForms.**
2. **Navrhněte možnosti využití těchto technologií v některém PHP Frameworku a implementujte je tak, aby výstup aplikace byl zobrazitelný na všech běžných web prohlížečích – včetně mobilních (Opera Mini atd.).**
3. **Na ukázkové aplikaci předveďte výhody použití těchto technologií ve srovnání s tradičním přístupem, kdy se musí aplikace programovat pro každou klientskou platformu zvlášť (tzn. aplikace generuje HTML pro desktopy, WML pro mobily).**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. YOUNG, Michael J. XML krok za krokem : 2. vydání. Brno : Computer press, 2006. 471 s.
2. ŽÁK, Miroslav. XML začínáme programovat . Praha : Grada Publishing, 2003. 200 s.
3. EISENBERG, David J. Using XForms with Mozilla. [s.l.] : O'Reilly Media, Inc., 2007. 46 s.
4. DUBINKO, Micah. XForms Essentials. [s.l.] : [s.n.], c2006. 215 s.

Vedoucí diplomové práce: **Ing. Tomáš Dulík**
Ústav aplikované informatiky

Datum zadání diplomové práce: **20. února 2009**

Termín odevzdání diplomové práce: **27. května 2009**

Ve Zlíně dne 13. února 2009

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Diplomová práce se zabývá možnostmi využití a nasazení technologie XForms, jenž je úzce spjatá s XML+XSLT, v PHP web aplikacích. Teoretická část stručně představuje technologie, jež jsou použity v části praktické, v níž jsem implementoval XForms do PHP frameworku Qcodo. Mým cílem bylo vytvořit nové knihovny frameworku, pomocí nichž budou programátoři schopni rychle a jednoduše vytvářet XForms formuláře pro své webové aplikace. V rámci praktické části jsem také vytvořil několik příkladových formulářů, jež demonstrují možnosti použití XForms a zabýval jsem se také podporou XForms a způsoby jejich zobrazení v předních internetových prohlížečích.

Klíčová slova:

XML, XSLT, XForms, PHP, framework, Qcodo, Qcodo XForms, FormsPlayer, FormFaces

ABSTRACT

This diploma thesis deals with the use and deployment of technology XForms, which is closely related to XML + XSLT in PHP web applications. The theoretical part briefly presents technologies that are used in practical part, in which I implemented XForms to PHP Framework Qcodo. My goal was to create a new library, through which the programmers will be able to quickly and easily create XForms forms for their web applications. Within the practical part I have also created several examples of forms, which demonstrate the practical possibilities of using XForms, and I also deal with XForms support in mainstream Internet browsers.

Keywords:

XML, XSLT, XForms, PHP, framework, Qcodo, Qcodo XForms, FormsPlayer, FormFaces

Chtěl bych poděkovat svému vedoucímu diplomové práce Ing. Tomáši Dulíkovi, především za velmi zajímavé téma, jež mi poskytl pro mou práci a také za vstřícný přístup, odborné vedení a čas, který mi věnoval.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.

Ve Zlíně …….

Podpis diplomanta

OBSAH

ÚVOD

Interakce, Web 2.0, e-commerce, prodej, zisk…to jsou nejčastěji skloňovaná slova dnešního Internetu. Internet se během své z hlediska dějin poměrně krátké existence rozvinul do gigantických rozměrů a nabyl velkého významu nejen v sociální, ale především také v ekonomické sféře. Vlastníci internetových stránek či portálů si uvědomují, že jejich webová prezentace musí umět zákazníkovi nabídnout daleko více než umožnila před několika lety. Musí mu nabídnout především interakci, přičemž základní komponentou, která uživateli umožňuje provádět tyto interaktivní operace, je formulář. Málokterý vlastník webových stran si však uvědomuje, jak komplikované je z hlediska webdesignu vytvářet inteligentní formuláře, které splňují prvky přístupnosti. Dnes stále nejčastěji používaná technologie pro webové formuláře, HTML Forms, totiž existuje již více než 14 let, jako dodatek do jazyka HTML verze 2.0, a v podstatě tak pamatuje vznik prvních internetových stránek. HTML formuláře tedy vznikly v době, kdy nikdo netušil, co vše se od nich bude v budoucnu očekávat. Bez více či méně složité aplikace dalších skriptovacích jazyků by formulář psaný v klasickém HTML jazyce neobstál ani v „nejzapadlejším“ internetovém obchodě.

Konsorcium W3C[1] se tedy pokusilo vyslyšet požadavky trhu a především vývojářů webových aplikací a již v roce 2003 představilo novou technologii XForms. Cesta vývoje formulářů budoucnosti, jež mají nahradit již nedostačující původní technologie, je však jak se zdá poměrně trnitá a zdlouhavá. Dnes je již k dispozici verze XForms 1.0 (Third Edition) jako doporučení W3C (vydáno 20.10.2007), verze XForms 1.1 má status Candidate Recommendation[2] a členové konsorcia slibují, že se doporučením stane v červnu 2009. Přestože XForms existují poměrně dlouho a chvíle, kdy se verze 1.1 stane standardem se blíží, jejich podpora ze stran internetových prohlížečů je stále nedostatečná. [1]

[1] W3C - World Wide Web Consortium, je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro WWW. Více: http://www.w3.org/.

[2] Candidate Recommendation je stav, kdy je daná technologie navržena na to stát se doporučením

Ve své diplomové práci jsem si dal za cíl vytvořit knihovny, jež vývojářům umožní snadnou, rychlou a efektivní tvorbu XForms formulářů a pomohou tak dostat tuto doposud poměrně neznámou technologii do povědomí tvůrců webových aplikací.

V teoretické části této práce budou stručně představeny technologie, které využívají XForms ke své funkci (XML, XSLT), ale i XForms samotné. Dále zde představím architekturu PHP frameworku Qcodo, jenž jsem si vybral pro implementaci XForms. Teoretická část si neklade za cíl předložit vyčerpávající specifikace jednotlivých technologií, jelikož jde o velmi obsáhlá témata, pouze předloží čtenáři stručné definice a především odkazy na další informační zdroje. Velmi často jsou zde jako zdroje uvedeny anglicky psané internetové stránky, jelikož jde o novou a aktuálně se vyvíjející problematiku a právě tyto zdroje nabízejí ty nejčerstvější informace.

Výsledkem praktické části mé diplomové práce je balík nových tříd pro PHP framework Qcodo pro tvorbu XForms formulářů. Jednotlivé třídy jsou dokumentovány ve formátu PHPDoc a vygenerovaná dokumentace (z důvodu publikování v komunitě Qcodo v anglickém jazyce) je součástí přílohy. V rámci praktické části jsou dále přiloženy konkrétní příkladové formuláře, jenž demonstrují použití vzniklé knihovny pro tvorbu XForms formulářů.

XForms se dnes stále dá ještě označit za technologii budoucnosti, která nabízí vývojářům nový přístup a nové možnosti v oblasti tvorby formulářů. A kdo jiný než právě vývojáři by se měl zasadit o to, aby tento velmi kvalitní produkt skupiny W3C získal co nejdříve podporu předních internetových prohlížečů a nahradil tak „zkostnatělé“ HTML formuláře.

I. TEORETICKÁ ČÁST

1 XML A XSLT

V této kapitole uvedu stručnou definici XML a XSLT a několik informačních zdrojů, jež se týkají této problematiky, jelikož detailní popis by vydal sám o sobě na další rozsáhlou práci.

1.1 Definice XML

Michael J. Young: *„XML (eXtensible Markup Language) je momentálně nejslibnější jazyk pro uchovávání a výměnu informací na webu.“* [2]

XML je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat.

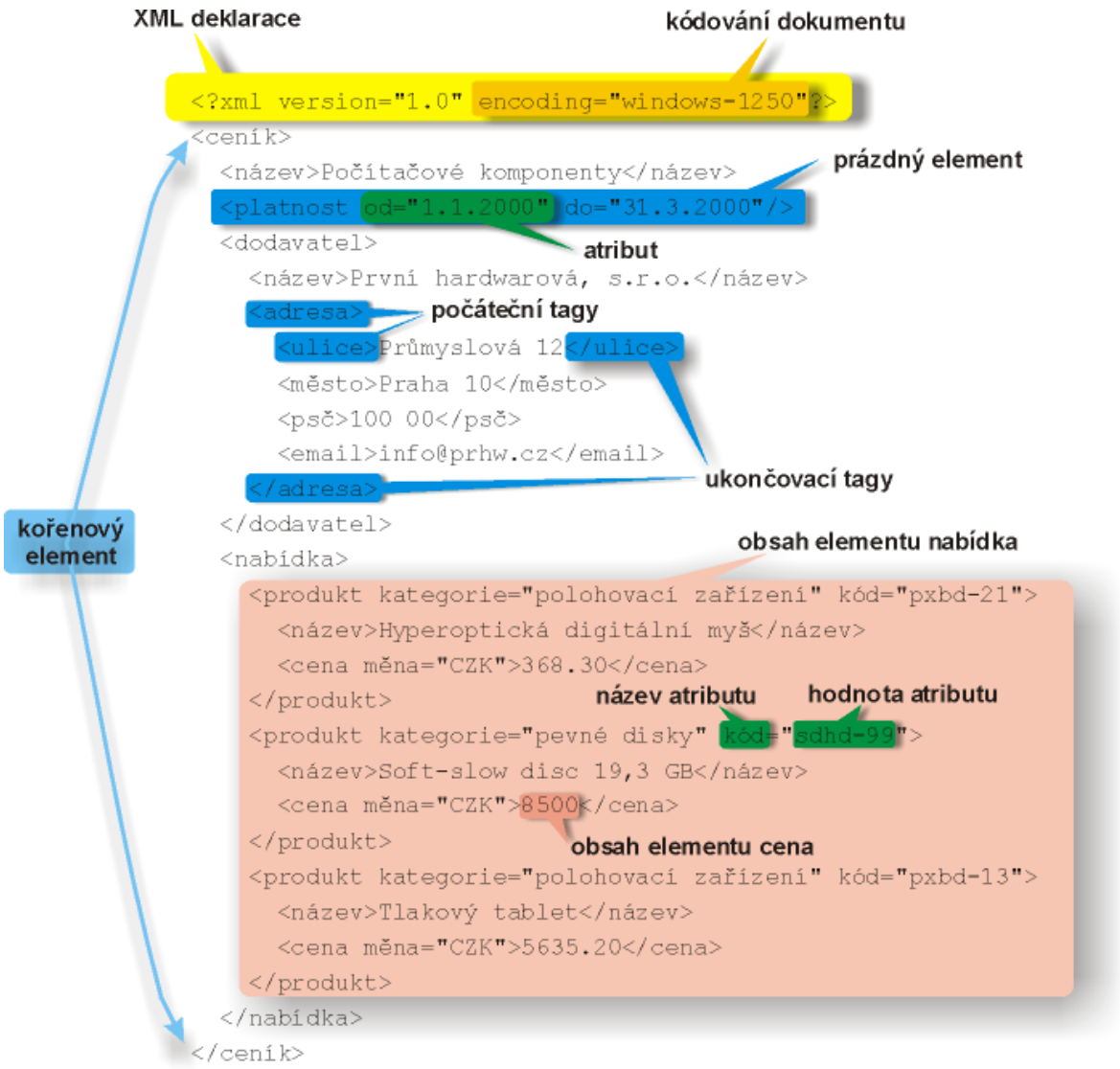
Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje pomocí kaskádových stylů. Další možností je pomocí XSLT stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML. [3]

1.2 Definice XSLT

Smyslem XSLT (eXtensible Stylesheet Language Transformations) je na základě zdrojového souboru a šablony vygenerovat jiný, třetí dokument nebo obecně soubor. Struktura tohoto výstupu XSLT není definována přímo standardem a je závislá na procesoru XSLT. Nejčastěji se používá výstup do HTML nebo XML, případně prostý textový formát, označovaný též TXT. Dalšími velmi známými výstupy jsou formáty PDF a RTF. Samozřejmě to však mohou být i libovolné jiné soubory nebo formáty dat. [4]

1.3 Syntaxe XML

Pro názorné zobrazení syntaxe jazyka XML zde uvádím ilustrační obrázek z knihy Jiřího Koska, „Základy jazyka WML“.



Obrázek 1: Syntaxe jazyka XML, zdroj [5]

Z obrázku je zřejmé, že každý XML dokument musí obsahovat deklaraci XML nejlépe včetně kódování. Jako každý značkovací jazyk používá i XML značky - tagy. Názvy těchto značek – tagů, jsou ovšem volitelné a záleží pouze na tvůrci XML dokumentu, jak kterou značku pojmenuje. Značky mohou být prázdné (jde pouze o jeden prázdný element), nebo

obsahují nějaký textový uzel a v tom případě jsou tvořeny počátečním a koncovým tagem. Každá značka může kromě svého obsahu, jenž je umístěn mezi počáteční a koncový tag, nést také další informace pomocí tzv. atributů. Ty se zapisují do těla počáteční značky.

1.4 XML Path Language (XPath)

XPath je jazyk určený pro adresování částí XML dokumentu.[6] Výrazy jazyka XPath vracejí uzly (nebo i více uzlů najednou), které odpovídají zadanému kritériu. [7]

Jednoduchý příklad:

XML:

```
1.  <shop>
2.      <product>
3.          <code>12345</code>
4.      </product>
5.  </shop>
```

XPath zvýrazněného uzlu je v tomto případě: "/shop/product/code".

1.5 Využití XML

XML jazyk má dnes velmi rozsáhlou škálu použití. Původně byl navržen pro značkování dokumentů textové povahy jako jsou knihy, technická dokumentace, články či webové stránky. Později se však projevila jeho univerzálnost a XML se prosadilo jako formát pro výměnu dat mezi různými informačními systémy.

1.5.1 XML jako prostředek obchodní komunikace

XML zaznamenalo velký úspěch právě v podnikových informačních systémech, jako formát pro výměnu obchodních dat. Stalo se velmi oblíbeným a využívaným formátem pro přenos obchodních či podnikových dat, jako jsou faktury, objednávky apod.

Pro tyto účely vzniklo několik XML knihoven, které standardizují formáty obchodní komunikace, jako je UBL – Universal Business Language společnosti OASIS. [http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl]

1.5.2 Aplikace XML

Formát XML se stal základem mnohých dalších značkovacích jazyků, které jsou z něj odvozeny a nazývají se aplikacemi XML. Níže uvádím několik příkladů aplikací XML.

DocBook XML – formát DocBook původně vznikl jako aplikace jazyka SGML, později se ale vyvinula přímo aplikace jazyka XML. Původě měl DocBook sloužit pro tvorbu hardwarové či softwarové dokumentace, ale dnes je používán jako prostředek pro zápis libovolných knih a článků. [http://www.docbook.org/]

MathML (Mathematical Markup Language) – vznikl aby pomohl vyřešit nedostatky HTML v oblasti zápisu matematických vzorců. [http://www.w3.org/Math/]

WML (Wireless Markup Language) – značkovací jazyk na bázi XML primárně určen mobilním úzkopásmovým zařízením. Využívá protokol WAP. [http://www.w3schools.com/WAP/wml_reference.asp]

XHTML (eXtensible HyperText Markup Language) – první specifikace jazyka vznikla s cílem převést starší jazyk HTML tak, aby vyhovoval podmínkám tvorby XML dokumentů a přitom byla zachována zpětná kompatibilita. [http://www.w3.org/TR/xhtml1/]

1.6 XML a XForms

Jak spolu tedy souvisejí technologie XML a XForms? XForms jsou také aplikací jazyka XML.

XForms jsou do XML integrovány: jsou psány v XML, data jež jsou získaná z formuláře jsou také ve formátu XML. XForms mohou načíst externí XML dokument jako inicializační data pro formulář a umí výsledek odeslat také jako XML dokument. Pokud je do tohoto XML soukolí vložen uživatel, má to za následek, že můžeme získat plnou XML vazbu od uživatele až k nám. [8]

2 XFORMS

V následující kapitole bude představena nová webová technologie XForms pro tvorbu webových formulářů. XForms vznikly proto, aby nahradily stávající zastaralé HTML formuláře a přestože jejich první verze XForms 1.0 je W3C doporučením již od roku 2003, nedošlo prozatím k jejich většímu rozšíření a podpora internetových prohlížečů je v současné době nedostatečná. Situaci by ovšem měl změnit příchod doporučení XHTML 2.0 jehož součástí již budou XForms 1.1, které by v jazyce XHTML měly definitivně nahradit HTML formuláře.

2.1 Hlavní výhody XForms

Poznámka: Následující text je volně přeložen ze zdroje [8].

XForms zvyšuje užitnost

XForms dávají okamžitou odezvu formuláře na vyplněná data a to bez nutnosti odeslání dat na server, či použití skriptovacích jazyků.

Je to XML a umožňuje odeslat XML

XForms jsou aplikací XML a stejně tak data odeslaná formulářem nebo inicializační data jsou ve formátu XML.

XForms je kombinací existujících technologií

XForms ke své funkcionalitě využívají již zaběhnutých a známých technologií, jako např. XML XPath (adresování a výpočet hodnot) a XML Schéma (definice datových typů).

Jsou nezávislé na platformě

Stejný formulář může být bez rozdílu dodán klasickému prohlížeči, PDA, mobilnímu telefonu či jiným klientům. Podmínkou je podpora XForms prohlížečem, pak není třeba jeden formulář poskytovat ve více verzích (např. HTML, WML).

Je jednodušší vytvořit komplikované formuláře

Implementace sofistikovaných formulářů, jež splňují nejvyšší standardy je díky XForms jednodušší a není potřeba jejich funkcionalitu řešit dalším skriptováním.

2.2 Co nabízejí XForms oproti původním HTML formulářům?

Poznámka: Následující text je volně přeložen ze zdroje [8].

XForms mají veškerou funkčnost klasických HTML formulářů a umožňují ještě více. Lze s nimi dělat následující

- Kontrolovat data, zatímco je uživatel vepisuje.
- Indikovat, že některá pole je nutné vyplnit a formulář proto bez nich nesmí být odeslán.
- Odesílat formulářová data v XML.
- Integrovat s webovými službami využívajícími SOAP či XML RPC.
- Odeslat jeden formulář několika různým serverům (například jeden hledaný řetězec několika vyhledávačům).
- Uložit hodnoty do souboru a obnovit je z něj.
- Použít data z odeslaného formuláře jako vstup jiného formuláře.
- Získat iniciační data pro formulář z externího dokumentu.
- Vypočítat odesílané hodnoty z různých formulářových polí.
- Různými způsoby omezovat hodnoty, jako např. tím, že je hodnota vyžadována, nebo musí být v určitém rozsahu.
- Vytvořit nákupní košík, nebo formulář ve stylu průvodce bez potřeby skriptování.

2.3 Konstrukce XForms formuláře

Tvorba XForms formuláře se v podstatě velmi liší od tvorby klasického HTML formuláře a koncept XForms je rozdělen na dvě části – *XForms Model* a samotné formulářové elementy (*Form Element*), jež jsou zobrazovány uživateli.

2.3.1 XForms Model

XForms Model popisuje data formuláře, jejich datové typy a způsob odesílání. Data jsou striktně oddělena od prezentace. V *XForms Modelu* jsou dále definovány tyto důležité

sekce: *Instance Element* (viz. 2.3.1.1), *Bindings* (viz. 2.3.1.2), *Submission Element* (viz. 2.3.1.3).

2.3.1.1 Instance Element

Instance Element definuje data, jež má formulář sbírat. Formuláře XForms vždy sbírají data pro XML dokument a jeho šablonu tvoří právě *Instance Element*. Je-li potřeba některá formulářová pole předvyplnit, musí být příslušná pole naplněna již v *Instance Elementu*.

Příklad *Instance Element*, pro formulář, v němž má uživatel vyplnit své jméno a příjmení:

```
1.  <osoba>
2.     <jmeno/>
3.     <prijmeni/>
4.  </osoba>
```

Potřebujeme-li formulářové pole předvyplnit, zajistíme to pomocí *Instance Elementu*:

```
1.  <osoba>
2.     <jmeno>Radek</jmeno>
3.     <prijmeni>Vala</prijmeni>
4.  </osoba>
```

2.3.1.2 Bindings

Anglické slovo *bindings* znamená česky *spoje* a právě v části *Bindings*, jež je součástí XForms Modelu, můžeme propojit konkrétní datový uzel z *Instance Elementu* s konkrétním formulářovým elementem, jež se bude zobrazovat uživateli. Navíc každému datovému uzlu z modelu můžeme přiřadit konkrétní datový typ nebo například to, zda je pole nutné vyplnit. Spoje se tvoří pomocí atributů *id* v *Binding Elementu* a *bind* ve formulářovém elementu (viz. Obrázek 2: Koncept XForms, zdroj [9]).

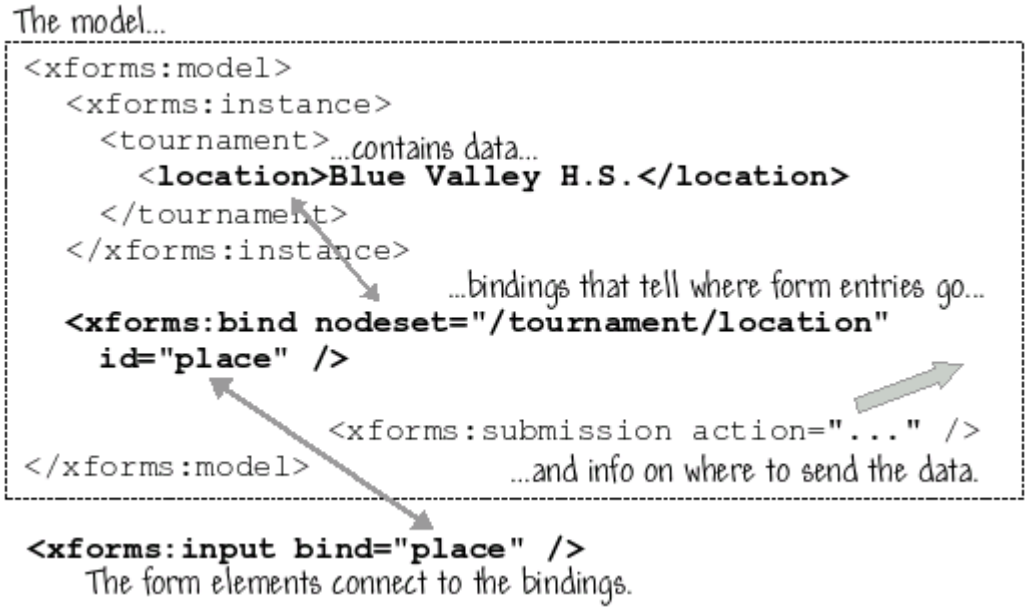
2.3.1.3 Submission Element

Poslední částí XForms Modelu je *Submission Element* – odesílací element. Na něm záleží kam a jakým způsobem se budou odesílat data z formuláře. Ekvivalentem *Submission Elementu* byly v HTML formulářích atributy tagu *form – action* a *method*. XForms však nabízí podstatně lepší funkcionalitu, jež umožní odeslat data například na několik různých míst, nebo je nechat zapsat do XML souboru přímo na disk.

2.3.2 Form Element

Form Element je konkrétní prvek formuláře, jež se zobrazuje uživateli a který je svázán s konkrétním datovým uzlem modelu. Může to být například textové pole – *input, textarea*, nebo menu – *select, select1*.

Následující obrázek z knihy J. Davida Eisenberga *Ussing XForms with Mozilla* [9] názorně ukazuje základní koncepci XForms.



Obrázek 2: Koncept XForms, zdroj [9]

2.4 Integrace XForms do webové stránky

2.4.1 MIME typ stránky s XForms

Jelikož XForms syntaxe je v podstatě XML, je nutné pro správnou interpretaci prohlížeči vkládat XForms do stránek, které mají MIME typ „application/xml+xhtml“. Toho lze docílit samozřejmě uložením do souboru *.xhtml, nebo například pomocí PHP funkce *header()* posílat správný MIME typ souboru.

2.4.2 Použití jmenných prostorů - namespace

Současná verze XHTML 1.0 vyžaduje pro XForms využívání namespace, ale nástupce XHTML 2.0 již má XForms přímo obsahovat, bez nutnosti použití jmenných prostorů. V XForms formulářích lze využít následující jmenné prostory:

http://www.w3.org/2002/xforms - Oficiální jmenný prostor pro XForms, v HTML a současné verzi XHTML 1.0 je potřebný pro deklarování všech XForms elementů .

http://www.w3.org/2001/XMLSchema - XForms zahrnuly datové typy definované ve W3C XML Schéma. Tento jmenný prostor se tedy využívá k definování datových typů ve formuláři.

http://www.w3.org/2001/xml-events - Pro zachycení událostí používají XForms také W3C schéma – XML events.

http://xforms.example/xforms/types - XForms schéma definující MIME typy souborů (pro formulářový element sloužící k uploadu souborů).

Praktické použití jmenných prostorů pro kompletní funkcionalitu XForms v XHTML dokumentu:

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2.  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3.  <html
4.  xmlns="http://www.w3.org/1999/xhtml"
5.  xmlns:xforms="http://www.w3.org/2002/xforms"
6.  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
7.  xmlns:ev="http://www.w3.org/2001/xml-events"
8.  xmlns:ftype="http://xforms.example/xforms/types">
9.  </html>
```

2.4.3 Definice modelu formuláře – XForms Model

XForms model (viz. 2.3.1) je XML reprezentací dat, jež by měl formulář získávat od uživatele, či výpočtem se zadaných hodnot a definuje také datové typy polí a způsob odeslání formuláře. Model se vkládá do XHTML stránky do sekce *head*.

Příklad XForms Modelu v XHTML stránce:

```
1.  <head>
2.  <title>XForms</title>
3.  <xforms:model id="o1">
4.      <xforms:instance>
```

```
5.            <shop xmlns="">
6.                <product>
7.                    <code />
8.                    <name />
9.                    <description />
10.               </product>
11.           </shop>
12.       </xforms:instance>
13.   <xforms:bind nodeset="/shop/product/code type="xsd:integer" />
14.   <xforms:bind nodeset="/shop/product/name />
15.   <xforms:bind nodeset="/shop/product/description" />
16.   <xforms:submission id="TestForm" method="post"
      includenamespaceprefixes="" omit-xml-declaration="true"
      action="action.php">
17.   </xforms:submission>
18.   </xforms:model>
19.   </head>
```

2.4.4 Základní GUI elementy v XForms

Samotnou uživatelskou grafiku vykresluje internetový prohlížeč, jež podporuje XForms[3]. Díky tomuto faktu by tedy XForms měly být naprosto nezávislé na prohlížeči i zařízení, na kterém běží. Nutnou podmínkou je však ona podpora XForms. Jak již bylo zmíněno, díky architektuře XForms jsou data oddělena od prezentace, a záleží pouze na tvůrci, jaká data či formulářová pole nechá vykreslit. Odpadá tak použití skrytých polí, jako v dřívějších HTML formulářích.

Níže je uveden výpis základních formulářových prvků XForms. Ty se zapisují do těla XHTML stránky mezi značky *body*.

[3] Všechny obrázky formulářů XForms (není-li uvedeno jinak) jsou renderovány z vlastních příkladů prohlížečem Mozilla Firefox vezre 3.0.10 s pluginem Mozilla XForms 0.8.6ff3.

2.4.4.1 XForms Label

K přiřazení popisů formulářových polí slouží v XForms značka *label*. Ta se vkládá mezi značky rodičovského prvku – tedy prvku formulářového pole, jež má *label* popisovat.

Příklad použití:

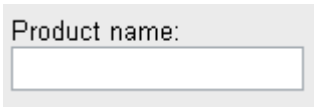
```
1.  <xforms:input ref="produkt_name">
2.      <xforms:label>Jméno produktu:</xforms:label>
3.  </xforms:input>
```

2.4.4.2 XForms Input

Značka *input* je známá již z HTML formulářů. V XForms má zajišťovat stejný účel – tedy vykreslit jednořádkové textové pole.

Příklad použití:

```
1.  <xforms:input bind="name">
2.      <xforms:label>Product name:</xforms:label>
3.  </xforms:input>
```



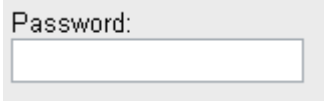
Obrázek 3: XForms Input

2.4.4.3 XForms Secret

Značka *secret*, vznikla nově pro XForms, v HTML formulářích bylo vykreslování textových polí pro zadání hesla řešeno pomocí atributu *type* s hodnotou *password*.

Příklad použití:

```
1.  <xforms:secret bind="password">
2.      <xforms:label>Password:</xforms:label>
3.  </xforms:secret>
```



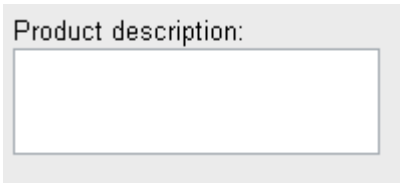
Obrázek 4: XForms Secret

2.4.4.4 XForms Textarea

Značka *textarea* má za úkol, stejně jako u HTML formulářů, vykreslit víceřádkové textové pole.

Příklad použití:

```
1.  <xforms:textarea bind="description">
2.      <xforms:label>Produkt description:</xforms:label>
3.  </xforms:textarea>
```



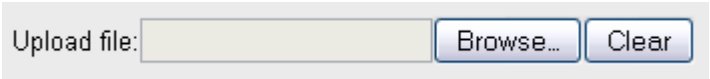
Obrázek 5: XForms Textarea

2.4.4.5 XForms Upload

Pro nahrávání souborů na web přes formulář, slouží v XForms značka *upload*. Ta nahrazuje původní *input* (z HTML formulářů) s atributem *type* nastaveným na hodnotu *file*.

Příklad použití:

```
1.  <xforms:upload bind="file">
2.     <xforms:label>Upload file:</xforms:label>
3.  </xforms:upload>
```



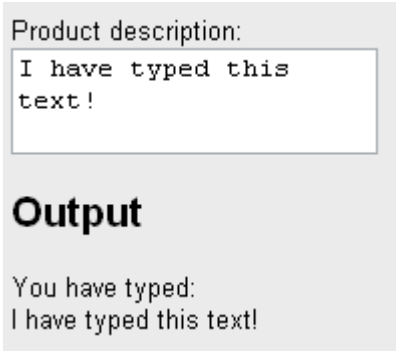
Obrázek 6: XForms Upload

2.4.4.6 XForms Output

XForms přinášejí proti původním HTML formulářům například možnost vypsat uživateli nějaký výstup, jež může být hodnota nějakého pole ve formuláři, nebo výsledek vypočítaný z hodnot. O tuto funkci se stará značka *output*. V příkladu níže je uveden jednoduchý výpis obsahu již zadaného uzlu /shop/produkt/description.

Příklad použití:

```
1.  <xforms:output ref="/shop/product/description">
2.      <xforms:label>You have typed:</xforms:label>
3.  </xforms:output>
```



Obrázek 7: XForms Output

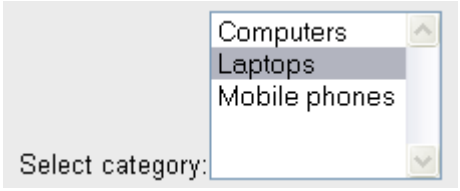
2.4.4.7 XForms Select1

Pro výběr 1 položky z výběrového menu je určen element *select1*. Vykreslení tohoto elementu lze ovlivnit parametrem *appearance*. Hodnotou může být, *minimal*, compact, *full*. Jednotlivé styly jsou vyobrazeny v obrázcích níže.

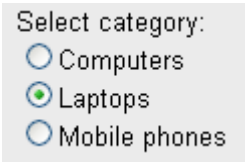
Příklad použití:

```
1.  <xforms:select1 bind="category">
2.     <xforms:label>Select category:</xforms:label>
3.     <xforms:item>
4.        <xforms:label>Computers</xforms:label>
5.        <xforms:value>1</xforms:value>
6.     </xforms:item>
7.     <xforms:item>
8.         <xforms:label>Laptops</xforms:label>
9.         <xforms:value>2</xforms:value>
10.    </xforms:item>
11.    <xforms:item>
12.        <xforms:label>Mobile phones</xforms:label>
13.        <xforms:value>1 </xforms:value>
14.    </xforms:item>
15. </xforms:select1>
```

Obrázek 8: XForms Select1, appearance: minimal



Obrázek 9: XForms Select1, appearance: compact



Obrázek 10: XForms Select1, appearance: full

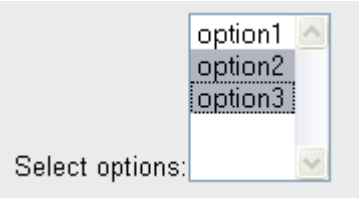
2.4.4.8 XForms Select

Pro vybrání jedné či více položek ze seznamu je určen element *select*. Vykreslení tohoto elementu lze ovlivnit parametrem *appearance*. Hodnotou může být, *minimal*, compact, *full*. Jednotlivé styly jsou vyobrazeny v obrázcích níže.

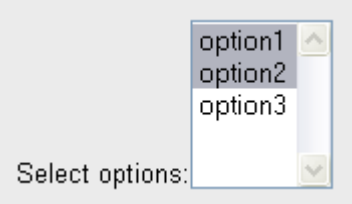
Příklad použití:

```
1.   <xforms:select bind="options">
2.       <xforms:label>Select options:</xforms:label>
3.        <xforms:item>
4.            <xforms:label>option1</xforms:label>
5.            <xforms:value>1</xforms:value>
6.         </xforms:item>
7.         <xforms:item>
8.            <xforms:label>option2</xforms:label>
9.            <xforms:value>2</xforms:value>
10.       </xforms:item>
11.       <xforms:item>
```

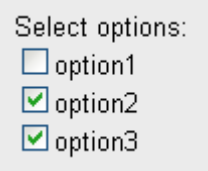
```
12.            <xforms:label>option3</xforms:label>
13.            <xforms:value>3 </xforms:value>
14.        </xforms:item>
15.  </xforms:select>
```



Obrázek 11: XForms Select, appearance: minimal



Obrázek 12: XForms Select, appearance: compact



Obrázek 13: XForms Select, appearance full

2.4.4.9 XForms Range

Nevšední formulářový prvek přináší element *range* (rozpětí). Prohlížeč by jej měl zobrazit jako posuvník, pomocí nějž uživatel vybere z předem definovaného rozpětí.

Příklad použití:

```
1.  <xforms:range bind="vote" start="1" step="1" end="5"
    incremental="true">
2.     <xforms:label>Vote:</xforms:label>
3.  </xforms:range>
```

Obrázek 14: XForms Range

2.4.4.10 XForms Trigger

K zachycení akcí slouží tlačítka, neboli *triggry*. Lze samozřejmě nastavit, na jakou událost má *trigger* reagovat a jakou akci má provést.

Jednoduchý příklad užití ukazuje tlačítko pro resetování formuláře:

```
1.  <xforms:trigger>
2.      <xforms:label>Reset</xforms:label>
3.      <xforms:reset ev:event="DOMActivate"></xforms:reset>
4.  </xforms:trigger>
```

Obrázek 15: XForms Reset

2.4.4.11 XForms Submit

Tlačítko pro odeslání formuláře se vykresluje pomocí značky *submit*. Ta obsahuje pouze atribut *submission*, jenž provazuje konkrétní tlačíko s konkrétním *Submission Elementem* v modelu formuláře. V *Submission Elementu* lze dále specifikovat způsob odeslání formuláře.

Příklad použití:

```
1.  <xforms:submit submission="BasicExample">
2.      <xforms:label>Odeslat</xforms:label>
3.  </xforms:submit>
```

Obrázek 16: XForms Submit

2.4.5 Pokročilé možnosti GUI v XForms

XForms nabízejí krom standardních formulářových prvků také další pokročilejší možnosti strukturování formuláře a to především díky elementům *switch*, *repeat* a *group*.

2.4.5.1 XForms Switch

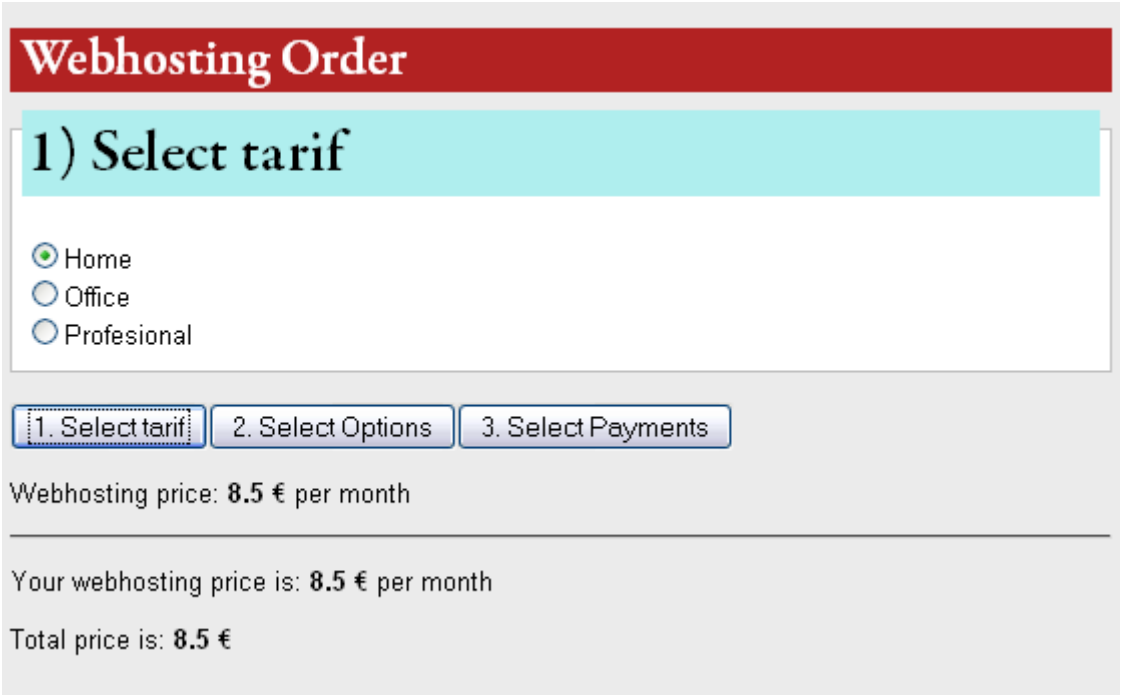
Pomocí elementu *switch* můžeme vytvořit například formulář ve stylu „průvodce“, nebo stránky sestávající se z několika přepínatelných záložek. V rodičovském elementu *switch* jsou zanořeni potomci – *case* – jednotlivé pohledy pro přepínání. Obsah elementu *case* je pak předložen prohlížeči, pokud na něj směřuje některé ze tlačítek – *trigger* – a je-li tato akce vyžádána.

Příklad použití:

```
1.  <xforms:switch>
2.      <xforms:case id="case1">
3.          <xforms:group>
4.              <!--content of case1->
5.          </xforms:group>
6.      </xforms:case>
7.     <xforms:case id="case2">
8.          <xforms:group>
9.              <!--content of case2->
10.         </xforms:group>
11.     </xforms:case>
12.     <xforms:case id="case3">
13.         <xforms:group>
14.             <!--content of case3->
15.         </xforms:group>
16.     </xforms:case>
17. </xforms:switch>
18. <xforms:trigger>
19.     <xforms:label>Case1</xforms:label>
```

```
20.       <xforms:toggle case="case1" ev:event="DOMActivate">
          </xforms:toggle>
21.   </xforms:trigger>
22.   <xforms:trigger>
23.       <xforms:label>Case2</xforms:label>
24.       <xforms:toggle case="case2" ev:event="DOMActivate">
25.       </xforms:toggle>
26.   </xforms:trigger>
27.   <xforms:trigger>
28.   <xforms:label>Case3</xforms:label>
29.       <xforms:toggle case="case3" ev:event="DOMActivate">
30.       </xforms:toggle>
31.   </xforms:trigger>
```

Výše uvedený kód pouze ilustruje použití elementu *switch*. Pro detailní pochopení jsem vytvořil názorný příklad (viz. *examples/webhosting.php*), více informací viz. 5.3.2



Obrázek 17: XForms příklad Switch

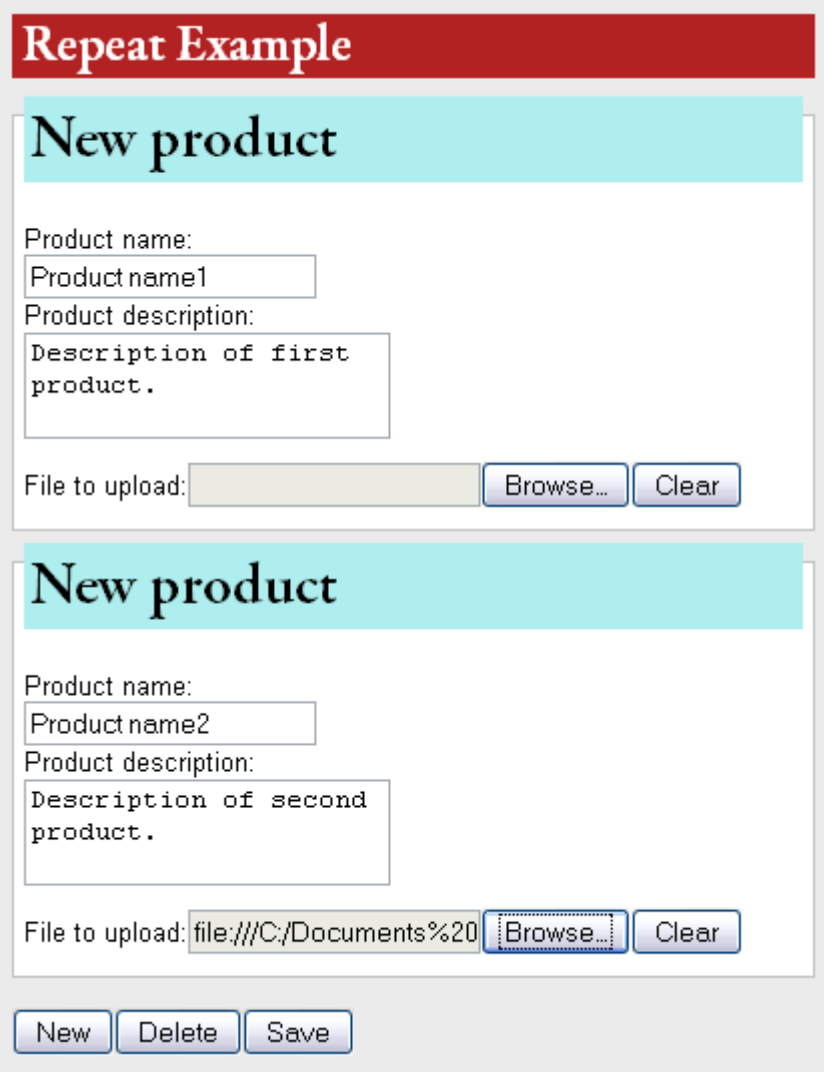
2.4.5.2 XForms Repeat

Element *repeat* umožňuje zopakovat libovolné vnořené elementy, pokud je tento požadavek zachycen tlačítkem. Umožňuje tak vytvořit například vícenásobné vkládání příloh k e-mailu, nebo třeba přidávání produktů, kde je možno bez odeslání dat přidat formulář pro vložení dalšího produktu.

Příklad použití:

```
1.  <xforms:repeat id="product-repeat" nodeset="product">
2.    <xforms:group><!--content to repeat--></xforms:group>
3.  </xforms:repeat>
4.  <xforms:trigger>
5.      <xforms:label>New</xforms:label>
6.      <xforms:insert nodeset="product" ev:event="DOMActivate"
    position="before" at="index('product-repeat')">
7.      </xforms:insert>
8.  </xforms:trigger>
9.  <xforms:trigger>
10.     <xforms:label>Delete</xforms:label>
11.     <xforms:delete nodeset="product[last()&gt;1]"
    ev:event="DOMActivate" at="index('product-repeat')">
12.     </xforms:delete>
13. </xforms:trigger>
```

Výše uvedený kód pouze ilustruje použití elementu *repeat*. Pro detailní pochopení jsem vytvořil názorný příklad (viz. *examples/repeat.php*), více informací viz. 5.2



Obrázek 18: XForms Repeat

3 FRAMEWORK QCODO

Qcodo je open-source vývojový framework, napsaný v PHP5. Tento framework je plně objektově orientovaný a poskytuje nástroje pro rychlý vývoj aplikací. Framework se skládá ze dvou hlavních částí: Generátor kódu a Qforms (třídy pro práci s HTML formuláři) a přestože byl vytvořen na vývoj větších projektů, lze jej využít jak v malých, tak i větších webových aplikacích.

Ve *Velkém testu PHP frameworků* [10], vyniklo Qcodo i svou rychlostí, když se zařadilo mezi nejrychlejší frameworky.

3.1 Základní vlastnosti Frameworku Qcodo

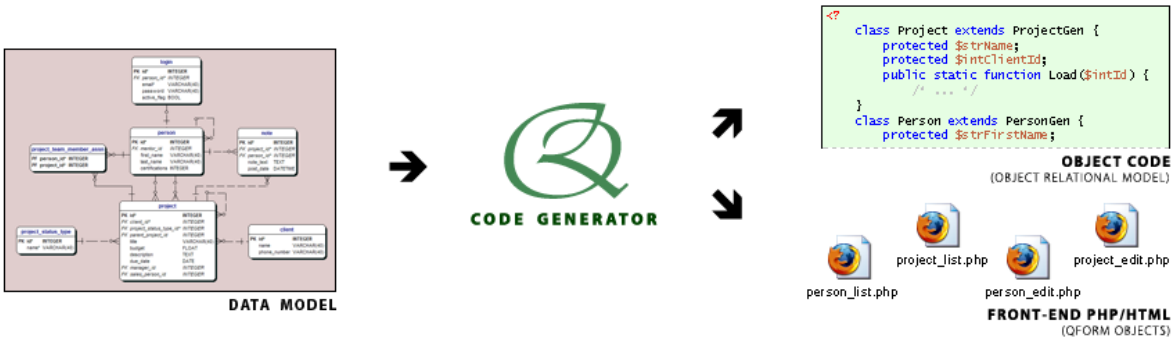
- Podporuje databáze MySQL, MSSQL a PostgreSQL.
- Do databáze přistupuje prostřednictvím ORM.
- Obsahuje moduly pro generování kódu. Ihned po instalaci a zobrazení souboru index.php máme možnost nechat si vygenerovat všechny modely, kontrolery a pohledy.
- Pohledy jsou tvořeny převážně PHP kódem doplněným HTML tagy.
- Obsahuje možnosti validace.
- Qcodo má velmi jednoduchou instalaci. Po rozbalení je potřeba nakonfigurovat soubor includes/configuration.inc.php, kde se nastavují cesty, url a databáze.

Informace převzaty ze zdroje [10].

3.2 Qcodo Code Generátor

Díky generátoru kódu může tvůrce webové aplikace obdržet ihned po generování automaticky vytvořené modely, kontrolery a pohledy, jež se generují na základě struktury databáze.

Následující obrázek ilustruje práci generátoru kódu:



Obrázek 19: Generátor kódu - Qcodo, zdroj [11]

II. PRAKTICKÁ ČÁST

4 IMPLEMENTACE XFORMS DO FRAMEWORKU QCODO

Jedním z předpokladů toho, že se nová technologie snadno prosadí, je jednoduchost jejího použití. XForms jsou dosti rozdílné od původních HTML formulářů a dosud nejsou moc známé a jejich nasazení tedy pro vývojáře znamená nutnost naučit se jejich syntaxi a specifika použití.

Cílem mé praktické části je přidat podporu XForms do PHP frameworku Qcodo, což umožní vývojářům generovat XForms formuláře přímo ze struktury databázových tabulek aplikace a používat je aniž by museli znát syntaxi XForms.

4.1 Proč právě Qcodo?

Na internetu je už poměrně velký výběr více či méně známých a kvalitních PHP frameworků. Je proto dobré stanovit si kritéria výběru.

Pro mě byla důležitá následující kritéria:

- Open-source framework psaný v PHP5
- Dostatečná rychlost frameworku
- Architektura MVC
- Generátor kódu, pro generování modelů, kontrolerů a pohledů přímo z databáze
- Dostatečně aktivní komunita kolem frameworku
- Jednoduchá a přímočará instalace frameworku

Tato všechna kritéria splňuje právě Qcodo a navíc obsahuje balík tříd QForms, jenž slouží ke zjednodušení práce s HTML formuláři, který na první pohled nabízel možnost inspirace, či využití.

4.2 Balík tříd QForms

První myšlenka po letmém prostudování architektury frameworku a komponenty QForms byla, využít stávající třídy QForms, které mají za úkol vykreslovat HTML formuláře, a pouze přidat možnost místo HTML formulářů vracet prohlížeči XForms. Po detailnějším prozkoumání architektury tříd a především podstatných odlišností syntaxe XForms od HTML, jsem však došel k názoru, že by to určitě nebylo ideální řešení. XForms jsou totiž

v určitých aspektech natolik odlišné, že pouhá změna *render* funkcí jednotlivých formulářových prvků by rozhodně nestačila a je potřeba vytvořit úplně odlišnou vykreslovací logiku pro kompletní formulář.

4.3 Balík tříd XForms

Rozhodl jsem se tedy vytvořit pro Qcodo kompletně nový balík tříd XForms, které budou určeny výhradně pro práci s XForms formuláři. Abych navázal na stávající zavedenou logiku konstrukce formuláře, jíž jsou vývojáři zvyklí užívat v Qforms, vytvořil jsem několik základních tříd pomocí nichž se tvoří objekty jednotlivých prvků formuláře, či formulář samotný.

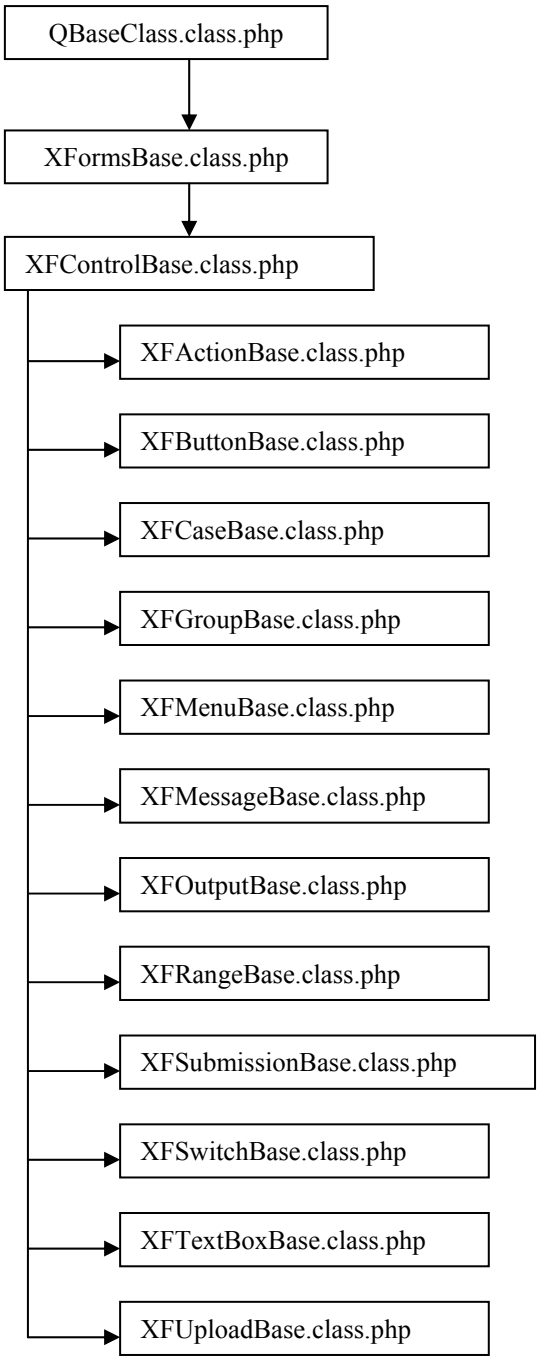
„Nejvyšší“ třídou vzniklého balíku je třída XForms (a její abstraktní třída XFormsBase). Obsahuje metody pro renderování formuláře a také metody pro automatickou tvorbu XML modelu formuláře, *Binding Elementů* a *Submission Elementu*.

Z této třídy dědí třída XFControl (a její abstraktní třída XFControlBase). Sdružuje metody pro nastavování parametrů, jež jsou společné více formulářovým elementům.

Jednotlivé prvky formuláře XForms pak mají svou vlastní třídu, jež extenduje třídu XFControl.

4.3.1 Schéma tříd XForms

Poznámka: Qcodo má ve svém jádře abstraktní třídy, jež obsahují klíčovou funkcionalitu a jsou dále děděny příslušnými třídami, jež může programátor libovolně upravit a implementovat vlastní funkčnost, aniž by zasahoval do funkčního jádra. Stejným způsobem jsem tvořil i třídy XForms a výpis výše je tedy výpisem abstraktních tříd. Třídy, které je extendují již zde vypsány nejsou, jejich struktura přirozeně odpovídá struktuře uvedených.



Obrázek 20: Schéma tříd modulu XForms

4.3.2 Načítání nových tříd v Qcodu

Ve frameworku Qcodo se o načítání potřebných tříd stará funkce Autoload, která je volána autoloaderem PHP5 (soubor QApplicationBase.class.php). Pro načtení nových tříd XForms bylo tedy nutné přidat do načítání i jejich umístění. Přidané řádky ve funkci Autoload jsou zvýrazněny tučným písmem:

```
1.  public static function Autoload($strClassName) {
2.   if (array_key_exists(strtolower($strClassName),
    QApplication::$ClassFile)) {
3.       require(QApplication::$ClassFile[strtolower($strClassName)]);
4.       return true;
5.   } else if (file_exists($strFilePath = sprintf('%s/%s.class.php',
    __INCLUDES__, $strClassName))) {
6.       require($strFilePath);
7.       return true;
8.   } else if (file_exists($strFilePath =
    sprintf('%s/qform/%s.class.php', __QCODO__, $strClassName))) {
9.       require($strFilePath);
10.      return true;
11.  } else if (file_exists($strFilePath =
    sprintf('%s/xforms/%s.class.php', __QCODO__, $strClassName))) {
12.      require($strFilePath);
13.      return true;
14.  } else if (file_exists($strFilePath =
    sprintf('%s/xforms/%s.class.php', __QCODO_CORE__, $strClassName))) {
15.      require($strFilePath);
16.      return true;
17.  }
18.  return false;
19.  }
```

4.3.3 Automatické generování XForms modelu

V kapitole 2.3.1 bylo řečeno, že každý XForms formulář musí v modelu obsahovat *Instance Element*, což je XML reprezentace dat formuláře. Při vytváření formuláře by si tvůrce samozřejmě měl promyslet, jaká data a v jaké struktuře bude formulář sbírat. Každý formulářový prvek tedy musí patřit určitému datovému uzlu XML modelu. Tento fakt mě přivedl na myšlenku automatického generování XML modelu. Psát ručně XML model každého formuláře může být totiž zdlouhavé a navíc se programátor může dopustit například takové chyby, že vytvoří nějaký datový uzel, k němuž nakonec nedefinuje žádný

formulářový prvek, nebo naopak vytvoří formulářový prvek, jenž nebude mít svůj datový uzel v modelu.

Třída *XFormsBase* tedy obsahuje funkci *GenerateXmlModel()*, jež se volá, není-li ručně zadán XML model jako řetězec. Při definování každého formulářového prvku je nutné zadat jeho *XPath* (viz. 1.4), jež je naparsována funkcí pro generování modelu a je z ní automaticky sestaven XML model formuláře. Pokud některý prvek formuláře nemá mít svůj datový ekvivalent v XML modelu, jeho *XPath* se jednoduše neuvádí (například tlačítka).

4.3.4 Generování formulářových elementů

Jednotlivé elementy se vykreslují do pohledu voláním funkce *Render()*. Ta však pouze zobrazí XML strukturu formulářového elementu, kterou konstruuje metoda *GetControlElement()* za pomocí dalších funkcí.

Práce s XML strukturou probíhá přes PHP XML DOM. Více informací o jednotlivých metodách naleznete v přiložené dokumentaci na CD.

4.3.5 Nastavování parametrů formulářových elementů

Přestože jsem se při tvorbě *XForms* knihoven snažil vycházet z architektury *Qcodo* frameworku a komponenty *QForm*, použil jsem nakonec místo magické funkce PHP *_set*(), funkci *_call*(), která umožňuje řetězit za sebe volání jednotlivých nastavovacích funkcí. Natavení vlastností textového pole pak může programátor napsat na jeden řádek, jelikož funkce *_call*() vždy vrací $*this*.

Příklad vytvoření a nastavení textového pole pomocí knihoven *XForms*:

```
1.  $this->ctrlProductName = new XFTextBox($this);
2.  $this->ctrlProductName->XPath('/shop/product/name')->Label('Product
    name:')->BindId('name');
```

Konstrukce XForms pomocí nových knihoven bude blíže popsána v kapitole 5.

5 PŘÍKLADY A MOŽNOSTI POUŽITÍ KNIHOVNY XFORMS

Následující kapitola obsahuje detailněji rozepsané dva příklady konkrétního použití nově vzniklé knihovny XForms pro Qcodo. První velmi jednoduchý příklad by měl stačit pro základní pochopení práce s knihovnou a druhý demonstruje složitější funkcionalitu. Pro podrobnější informace je k dispozici dokumentace API.

5.1 Jednoduchý formulář pro vyhledávání

První příklad demonstruje vytvoření velmi jednoduchého formuláře pro zadání hledaného výrazu.

5.1.1 Kontroler

Výpis zdrojového kódu souboru search.php:

```
3.  <?php
4.   require('../includes/prepend.inc.php');
5.   class SearchForm extends XForms {
6.      protected $ctrlSearch;
7.      protected $btnSearch;
8.
9.      public function Form_Create() {
10.        $this->ctrlSearch = new XFTextBox($this);
11.        $this->ctrlSearch->XPath('/search/string')->Label('String to
    search:')->BindId('string');
12.
13.        $this->btnSearch = new XFButton($this);
14.        $this->btnSearch->Submit('self')->Label('Save');
15.      }
16. }
17. $xmlModel = '';
18. SearchForm::Run('SearchForm', 'search.tpl.php', $xmlModel);
19. ?>
```

Nejprve je nutné načíst soubor *prepend.inc.php*, který zprostředkovává veškerou funkčnost frameworku QCodo (řádek 4). Dále se vytvoří nová třída formuláře, v našem případě SearchForm, (řádek 5) a v ní definujeme proměnné *$ctrlSearch* a *$btnSearch*, jež budou obsahovat objekty jednotlivých prvků formuláře. V metodě Form_Create jsou pak výše zmíněné objekty vytvořeny a nastaveny. *$ctrlSearch* je textové pole – instance třídy *XFTextBox*. Na řádku 11 jsou mu nastaven parametr *XPath* (přesné určení uzlu v XML modelu), z nějž se zároveň XML model bude automaticky generovat. Dále pak *Label*, jež nastaví textový popisek pole a parametr *BindId*, jež nastaví identifikátor provázání do *Binding Elementu*. Dále je vytvořena instance třídy XFButton, která bude sloužit jako tlačítko pro odeslání formuláře (řádek 14). Nastavení *Submit* na hodnotu „self" zajistí to, že formulář pro odeslání použije implicitní *Submission Element*. XForms však umožňují odeslat jeden formulář více způsoby a právě nastavením *Submit* na jinou hodnotu než „self", můžeme zadat konkrétní identifikátor *Submission Elementu* jež se má použít pro odeslání (samozřejmostí je předpoklad, že *Submission Element* s příslušným ID ve formuláři vytvoříme). Vlastnost *Label* je identická pro všechny formulářové prvky a nastavuje jejich textový popis. Řádek 17 definuje řetězex *$xmlModel* jako prázdný, což má za následek automatické generování modelu z příslušných objektů formuláře. Na řádku 18 je volána statická metoda *Run()* třídy *XForms*. Jejímy parametry jsou název třídy vytvářeného formuláře, dále název souboru pohledu a již zmíněný *$xmlModel*.

5.1.2 Pohled

Výpis zdrojového kódu souboru *search.tpl.php*:

```
1.  <?php header("Content-Type: application/xhtml+xml; charset=UTF-8");
    ?>
2.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3.            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4.  <html xmlns="http://www.w3.org/1999/xhtml"
5.  xmlns:xforms="http://www.w3.org/2002/xforms"
6.  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
7.  xmlns:ev="http://www.w3.org/2001/xml-events"
8.  xmlns:ftype="http://xforms.example/xforms/types"
9.  xmlns:html="http://www.w3.org/1999/xhtml">
```

```
10.  <!-- watch this space -->
11.  <head>
12.  <title>XForms in QCodo - Repeat Example</title>
13.  <style type="text/css">@import url("<?php _p(__VIRTUAL_DIRECTORY__ .
     __CSS_ASSETS__); ?>/xforms_styles.css");</style>
14.  <?php $this->RenderBegin(); ?>
15.  </head>
16.  <body>
17.  <h1>Repeat Example</h1>
18.  <p><?php $this->ctrlSearch->Render(); ?></p>
19.  <p><?php $this->btnSearch->Render(); ?></p>
20.  <?php $this->RenderEnd(); ?>
21.  <?php require (__INCLUDES__ . '/footer.inc.php'); ?>
```

Na prvním řádku souboru pohledu je volána PHP funkce header(), jež zajistí to, že formulář bude prohlížeči zaslán s MIME typem application/xml+xhtml, který je při použití XForms vyžadován. V hlavičce dále můžeme vidět na řádcích 4 až 9 potřebné jmenné prostory (viz. 2.4.2). Důležitá je změna umístění volání metody *RenderBegin()*, která je při použití *QForm* komponenty v *Qcodu* až za XHTML tagem *head*. Třída *XForms* ale pomocí této metody vkládá *XForms Model*, jenž musí být obsažen v hlavičce, proto je metoda volána ještě před ukončením tagu *head*. V těle XHTML stránky jsou pak již volány funkce *Render()* příslušných objektů formuláře, jež zajistí vykreslení objektu na stránku. Na konec je volána funkce *RenderEnd()* a vložena patička stránky.

5.1.3 Výsledný zdrojový kód

Výsledný zdrojový kód, který obdrží prohlížeč vypadá následovně, tučně je označen kód, jež je vygenerován díky novému balíku tříd *XForms*:

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2.            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3.  <html xmlns="http://www.w3.org/1999/xhtml"
4.  xmlns:xforms="http://www.w3.org/2002/xforms"
5.  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6.  xmlns:ev="http://www.w3.org/2001/xml-events"
```

```
7.  xmlns:ftype="http://xforms.example/xforms/types"
8.  xmlns:html="http://www.w3.org/1999/xhtml">
9.  <!-- watch this space -->
10. <head>
11. <title>XForms in QCodo - Repeat Example</title>
12. <style type="text/css">@import
    url("/vyvoj/codoxforms/assets/css/xforms_styles.css");</style>
13. <xforms:model id="o1">
14.    <xforms:instance>
15.       <search xmlns="">
16.         <string></string>
17.       </search>
18.    </xforms:instance>
19.    <xforms:bind nodeset="/search/string" id="string"></xforms:bind>
20.    <xforms:submission id="SearchForm" method="post"
    includenamespaceprefixes="" omit-xml-declaration="true"
    action="/vyvoj/codoxforms/drafts/search.php"></xforms:submission>
21. </xforms:model>
22. </head>
23. <body>
24.    <h1>Repeat Example</h1>
25.    <p><xforms:input bind="string">
26.       <xforms:label>String to search:</xforms:label>
27.    </xforms:input></p>
28.    <p><xforms:submit submission="SearchForm">
29.       <xforms:label>Save</xforms:label>
30.    </xforms:submit></p>
31. </body>
32. </html>
```

Následovně může zobrazit formulář prohlížeč s podporou XForms:

Obrázek 21: Příklad vyhledávácího formuláře v XForms

5.2 Formulář s XForms Repeat

XForms Repeat je prvek jenž nemá v HTML formulářích obdoby. Pomocí elementu *Repeat* můžeme libovolněkrát zopakovat, či zkopírovat jeho obsah. Níže uvádím názorný příklad formuláře, jež je určen pro vkládání zboží do skladu. Uživatel ale nemusí po každé zadané položce formulář odesílat, ale může pokračovat v zadávání další položky.

5.2.1 Kontroler

Výpis zdrojového kódu souboru repeat.php.

```
1.  <?php
2.  require('../includes/prepend.inc.php');
3.
4.  class TestForm extends XForms {
5.   protected $ctrlProductName;
6.   protected $ctrlProductDescription;
7.   protected $ctrlUpload;
8.   protected $ctrlGroup;
9.   protected $btnNew;
10.  protected $btnDelete;
11.  protected $ctrlRepeat;
12.  protected $btnSave;
13.
14.  public function Form_Create() {
15.   $this->ctrlProductName = new XFTextBox($this);
```

```
16.    $this->ctrlProductName->XPath('/shop/product/name')-
     >Label('Product name:')->Ref('name')->Incremental(true);
17.    $this->ctrlProductDescription = new XFTextBox($this);
18.    $this->ctrlProductDescription->XPath('/shop/product/description')-
     >Label('Product description:')->TextMode(QTextMode::MultiLine)-
     >Ref('description');
19.    $this->ctrlUpload = new XFUpload($this);
20.    $this->ctrlUpload->XPath('/shop/product/upload')->Ref('upload')-
     >Label('File to upload:')->DataType('base64Binary');
21.    $this->ctrlGroup = new XFGroup($this);
22.    $this->ctrlGroup->Label('New product')->Append($this-
     >ctrlProductName,$this->ctrlProductDescription,$this->ctrlUpload);
23.    $this->ctrlRepeat = new XFRepeat($this);
24.    $this->ctrlRepeat->Id('product-repeat')->Nodeset('product')-
     >Append($this->ctrlGroup);
25.    $this->btnNew = new XFButton($this);
26.    $this->btnNew->Label('New')->InsertNodeset('product')-
     >InsertNodesetPosition('before')->InsertRepeatAt("index('product-
     repeat')");
27.    $this->btnDelete = new XFButton($this);
28.    $this->btnDelete->Label('Delete')-
     >DeleteNodeset('product[last()>1]')->DeleteRepeatAt("index('product-
     repeat')");
29.    $this->btnSave = new XFButton($this);
30.    $this->btnSave->Submit('self')->Label('Save');
31.   }
32.  }
33.  $xmlModel = '';
34.  TestForm::Run('TestForm', 'repeat.tpl.php', $xmlModel);
35.  ?>
```

Ve výše uvedeném kódu budu již komentovat pouze důležité části. Logika kódu totiž koresponduje s prvním jednoduchým příkladem viz. 5.1.1.

V tomto příkladě stojí za povšimnutí element *Group* (řádek 21) a jeho metoda *Append()* (řádek 22), pomocí níž se připojují potomci. Jestliže tedy chceme seskupit několik

formulářových prvků do jedné skupiny – elementu G*roup* – provedeme to výše uvedeným způsobem.

Na řádku 23 se vytváří nová instance třídy *XFRepeat*, jež do níž se pak připojí elementy, jež budeme chtít ve formuláři opakovat – kopírovat. Každý element *Repeat* musí mít svůj identifikátor, na který se poté propojují akce tlačítek, jež mají vyvolat kopírování, či smazání elementu *Repeat* a jeho potomků. Ten je nastaven na řádku 24 na hodnotu „product-repeat“. Na témže řádku je také pomocí metody *Append()* připojena skupina formulářových prvků pro opakování.

Na řádku 26 je pak několik důležitých kroků pro nastavení tlačítka „New“, jež po stlačení zkopíruje sekci *repeat*. Pro nastavení jsou důležité tří metody: *InsertNodeset()* – nastavuje *XForms* element *Input* a jeho atribut *nodeset*; *InsertNodesetPostion() – nastavuje* atribut *position; InsertRepeatAt()* slouží k nastavení atributu *at*.).

Na řádku 28 je pak nastavení tlačítka „Delete“ pro smazání vložené skupiny elementů. K nastavení se analogicky používají metody *DeleteNodeset()* a *DeleteRepeatAt()*. První opět definuje XPath cestu elementu, a druhý přesný uzel, který má být smazán.

Více informací o používání *XForms Repeat* zejména ohledně nastavení atributů je na stránkách W3C - XForms for HTML Authors, Part 2 (http://www.w3.org/MarkUp/Forms/2006/xforms-for-html-authors-part2.html#Repeat).

5.2.2 Pohled

Výpis zdrojového kódu souboru switch.tpl.php.

```
1.  <?php header("Content-Type: application/xhtml+xml; charset=UTF-8");
2.  ?>
3.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
4.            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
5.  <html xmlns="http://www.w3.org/1999/xhtml"
6.  xmlns:xforms="http://www.w3.org/2002/xforms"
7.  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
8.  xmlns:ev="http://www.w3.org/2001/xml-events"
9.  xmlns:ftype="http://xforms.example/xforms/types"
10. xmlns:html="http://www.w3.org/1999/xhtml">
```

```
11.
12.  <!-- watch this space -->
13.  <head>
14.  <title>XForms in QCodo - Repeat Example</title>
15.  <style type="text/css">@import url("<?php _p(__VIRTUAL_DIRECTORY__ .
     __CSS_ASSETS__); ?>/xforms_styles.css");</style>
16.  <?php $this->RenderBegin(); ?>
17.  </head>
18.  <body>
19.    <h1>Repeat Example</h1>
20.      <p><?php $this->ctrlRepeat->Render(); ?></p>
21.      <p><?php $this->btnNew->Render(); ?>
22.      <?php $this->btnDelete->Render(); ?>
23.      <?php $this->btnSave->Render(); ?></p>
24.
25.  <?php $this->RenderEnd(); ?>
26.  <?php require (__INCLUDES__ . '/footer.inc.php'); ?>´
```

V tomto příkladu je vidět, že krom tlačítek se volá *Render()* funkce pouze pro element *Repeat*. Ten již totiž obsahuje ostatní objekty formulářových polí, jež mu byly přiřazeny funkcí *Append()*.

5.2.3 Výsledný zdrojový kód

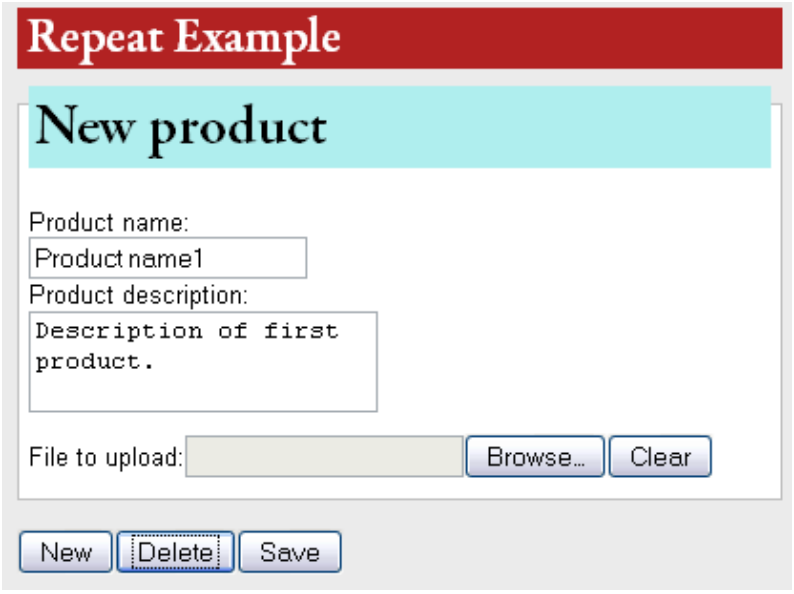
Výsledný zdrojový kód, který obdrží prohlížeč vypadá následovně, tučně je označen kód, jež je vygenerován díky novému balíku tříd *XForms*:

```
1.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2.            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3.  <html xmlns="http://www.w3.org/1999/xhtml"
4.  xmlns:xforms="http://www.w3.org/2002/xforms"
5.  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6.  xmlns:ev="http://www.w3.org/2001/xml-events"
7.  xmlns:ftype="http://xforms.example/xforms/types"
8.  xmlns:html="http://www.w3.org/1999/xhtml">
```

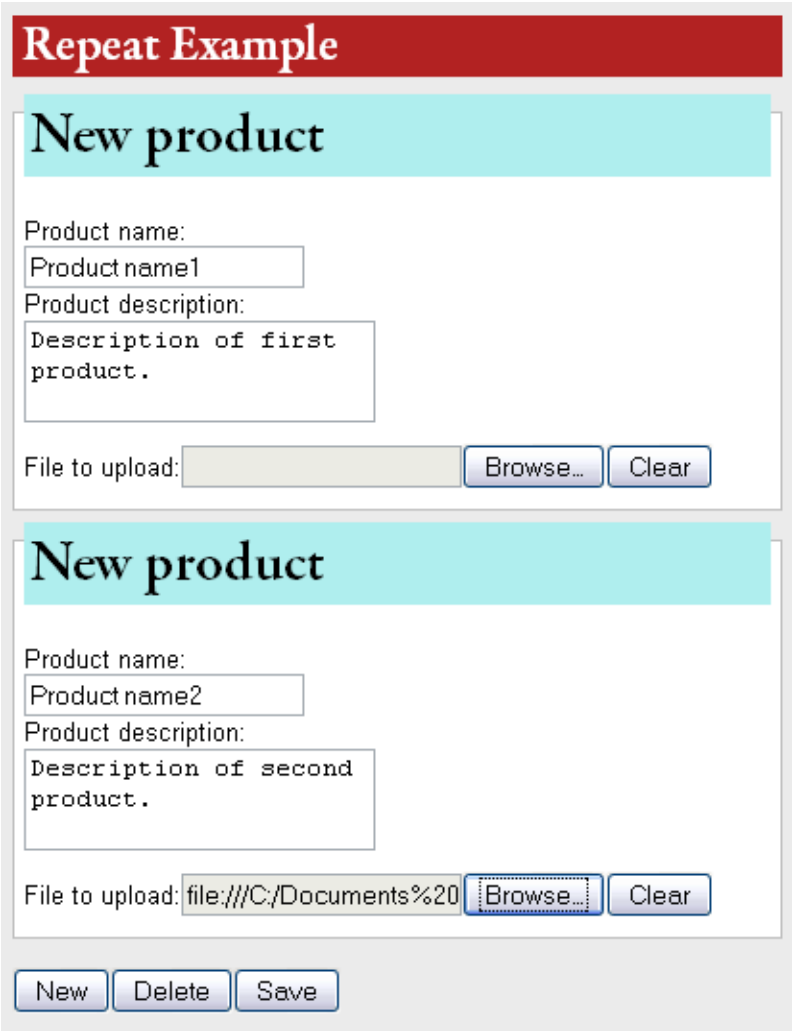
```
9.
10. <!-- watch this space -->
11. <head>
12. <title>XForms in QCodo - Repeat Example</title>
13. <style type="text/css">@import
    url("/vyvoj/codoxforms/assets/css/xforms_styles.css");</style>
14. <xforms:model id="o1">
15.    <xforms:instance>
16.     <shop xmlns="">
17.        <product>
18.           <name></name>
19.           <description></description>
20.           <upload></upload>
21.        </product>
22.      </shop>
23.    </xforms:instance>
24.    <xforms:bind nodeset="/shop/product/name"></xforms:bind>
25.    <xforms:bind nodeset="/shop/product/description"></xforms:bind>
26.    <xforms:bind nodeset="/shop/product/upload"
    type="xsd:base64Binary"></xforms:bind>
27.    <xforms:submission id="TestForm" method="post"
    includenamespaceprefixes="" omit-xml-declaration="true"
    action="/vyvoj/codoxforms/drafts/repeat.php"></xforms:submission>
28. </xforms:model>
29. </head>
30. <body>
31. <h1>Repeat Example</h1>
32. <xforms:repeat id="product-repeat" nodeset="product">
33.    <xforms:group>
34.       <xforms:label>New product</xforms:label>
35.       <xforms:input ref="name" incremental="true">
```

```
36.            <xforms:label>Product name:</xforms:label>
37.         </xforms:input>
38.         <xforms:textarea ref="description">
39.            <xforms:label>Product description:</xforms:label>
40.         </xforms:textarea>
41.         <xforms:upload ref="upload">
42.            <xforms:label>File to upload:</xforms:label>
43.         </xforms:upload>
44.      </xforms:group>
45.   </xforms:repeat>
46.   <p><xforms:trigger>
47.      <xforms:label>New</xforms:label>
48.      <xforms:insert nodeset="product" ev:event="DOMActivate"
      position="after" at="index('product-repeat')">
49.      </xforms:insert>
50.   </xforms:trigger>
51.   <xforms:trigger>
52.      <xforms:label>Delete</xforms:label>
53.      <xforms:delete nodeset="product[last()&gt;1]"
      ev:event="DOMActivate" at="index('product-repeat')">
54.      </xforms:delete>
55.   </xforms:trigger>
56.   <xforms:submit submission="TestForm">
57.      <xforms:label>Save</xforms:label>
58.   </xforms:submit>
59.   </p>
60.   </body>
61.   </html>
```

Prohlížeč s podporou XForms může zobrazit výše uvedený zdrojový kód následovně:



Obrázek 22:XForms Repeat



Obrázek 23: XForms Repeat po zmáčknutí tlačítka New

Obrázek 24: XForms Repeat po zmáčknutí tlačítka Delete (kurzor nastaven na první produkt)

5.3 Další příklady

V rámci praktické části jsem vytvořil více příkladů použití XForms, ale jelikož logika konstrukce formulářů je stále stejná, nebudu je zde dále detailně popisovat. K dispozici jsou v adresáři examples.

5.3.1 Základní XForms elementy

Příklad, který obsahuje základní XForms elementy (*input, secret, textarea, output, upload, select, select1, range, trigger, submit*) naleznete v souboru *xfControlsExample.php*, k němuž náleží pohled *xfControlsExample.tpl.php*.

5.3.2 XForms Switch – nabídka webhostingových služeb

Demonstraci elementu *XForms Switch* obsahuje příklad formuláře webhostingové společnosti, která nabízí své služby. Naleznete jej v souboru *webhosting.php* (*webhosting.tpl.php*).

5.4 ORM Code Generátor

Framework Qcodo disponuje vlastním ORM generátorem kódu (viz. 3.2), což je jedna z hlavních funkcí, jež pomáhají při rychlém vývoji aplikací. Z databáze tak může nechat vývojář ihned vygenerovat ne jen třídy odpovídající jednotlivým tabulkám databáze, jež nabídnou základní CRUD funkcionalitu, ale i jednouché „drafty“ – formuláře, které umožní editaci údajů v databázi.

Code generátor frameworku Qcodo úzce spolupracuje s komponentou QForm, jež vytvoří zmíněné HTML formuláře, připomínající jednoduchou administraci databáze, z níž byl kód generován. Formuláře jsou generované pomocí šablon, jež je možné změnit tak, aby místo objektů QForm volaly objekty nově vzniklé třídy XForms.

Změnil jsem tedy šablony pro generování formulářů tak, aby po spuštění code generátoru byly vygenerovány drafty, používající XForms formuláře. Tato funkcionalita může být klíčová, protože podstatně zefektivní práci a může také sloužit jako výukový nástroj, kdy jsou na základě databázového modelu vygenerovány kontrolery i pohledy a programátor tak může projít jejich zdrojové kódy a využít je i pro „ruční“ tvorbu vlastních formulářů.

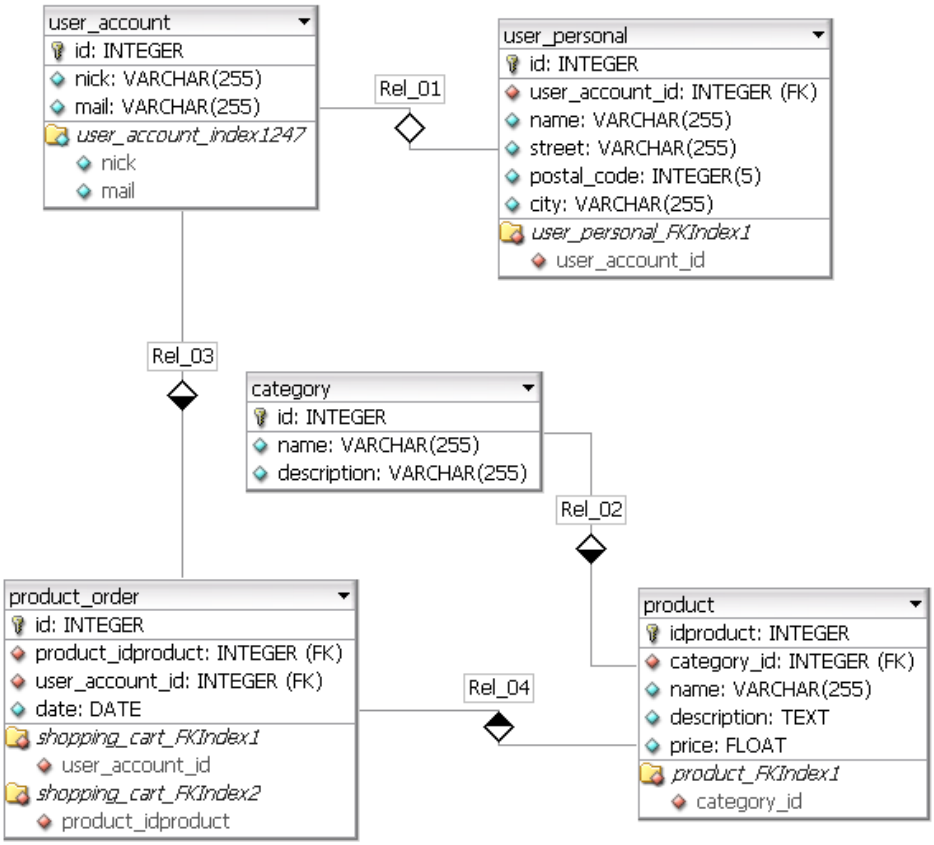
5.4.1 Příklad na automatické generování kódu

5.4.1.1 Databázový model

Mějme SQL databázi fiktivního objednávkového systému. Databázové tabulky jsou typu InnoDB, aby bylo možné plně využít i relační vztahy.

SQL kód databáze naleznete v příloze P1.

Model databáze vypadá následovně:



Obrázek 25: Model databáze

5.4.1.2 Vygenerované třídy

Code generátor frameworku Qcodo vygeneruje na základě databázového modelu (Obrázek 25) třídy, jež poskytují základní CRUD funkcionalitu k tabulkám. Třídy dokonce zohledňují relační vztahy. Více informací o generování na oficiálních stránkách projektu Qcodo [http://www.qcodo.com].

5.4.1.3 Vygenerované drafty

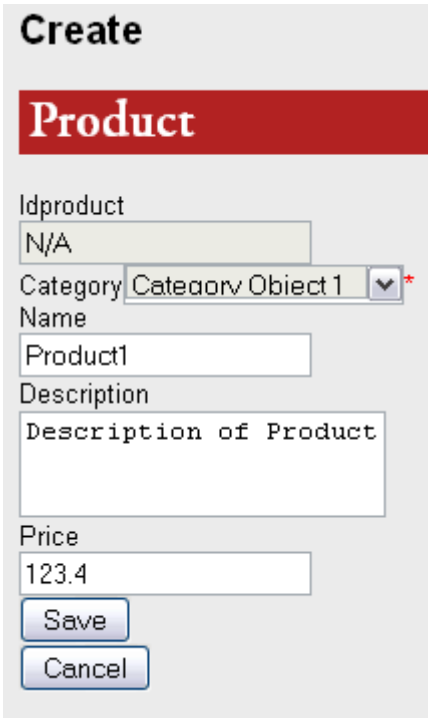
Draft, neboli návrh je tedy GUI vrstvou, jež umožňuje přímo editovat záznamy v databázi a generování draftů vychází z databázového modelu (Obrázek 25).

Díky úpravě šablon, kterou jsem provedl, generuje nyní code generátor XForms editační formuláře draftů.

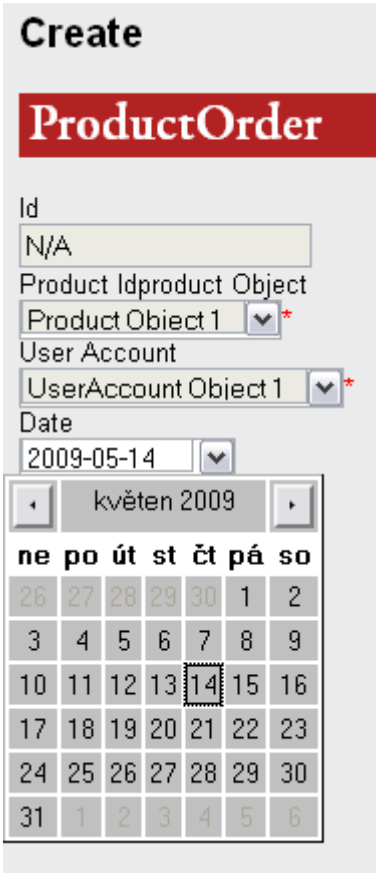
Níže uvádím ukázky některých draftů po vygenerování:

Obrázek 26: Ukázka formuláře pro vytvoření nové kategorie

Obrázek 27: Ukázka formuláře pro editaci kategorie



Obrázek 28: Ukázka formuláře pro vytvoření nového produktu



Obrázek 29: Ukázka formuláře pro vytvoření nové objednávky

6 PODPORA XFORMS V PRAXI

Přestože XForms jsou nesporně vynikající technologií, jež vývojářům v mnohém usnadní práci a staré HTML formuláře překonávají ve všech aspektech, na jejich nástup do praxe se stále ještě čeká. Hlavním důvodem je nedostatečná podpora XForms v dnešních internetových prohlížečích. V současnosti (květen 2009) nepodporuje ani jeden z nejrozšířenějších internetových prohlížečů (Internet Explorer, Mozilla Firefox, Opera) XForms v základní instalaci. Situace se samozřejmě řešit dá, ale pouze za použití různých pluginů, nebo technologií, jež XForms transformují přímo na serveru.

Tento stav samozřejmě způsobuje to, že XForms stále čekají na chvíli, kdy prohlížeče začnou podporovat XHTML verze 2.0 (jehož budou součástí) a do té doby zůstávají pouze výsadou některých uzavřených firemních systémů. Jejich nasazení například na e-shop je v současné době nesmysl, protože běžný uživatel si v současnosti nebude schopen formulář jednoduchou cestou zobrazit.

Ve chvíli kdy ale ona podpora prohlížečů přijde (XHTML 2.0 se standardem stane nejspíše v srpnu roku 2009 a jeho podpora by tedy na sebe neměla nechat dlouho čekat), budou moci vývojáři těžit ze všech přínosů nové formulářové technologie a rychle ji uvést do praktického používání.

Následující kapitoly popisují současné (květen 2009) možnosti zobrazení XForms v internetových prohlížečích.

6.1 Desktopové prohlížeče

6.1.1 XForms a Internet Explorer

Internet Explorer ani ve verzi 8 neumí zobrazit soubor s MIME typem application/xhtml+xml. To je pro XForms samozřejmě velký problém, jelikož soubor, který XForms obsahuje by měl mít právě tento MIME typ.

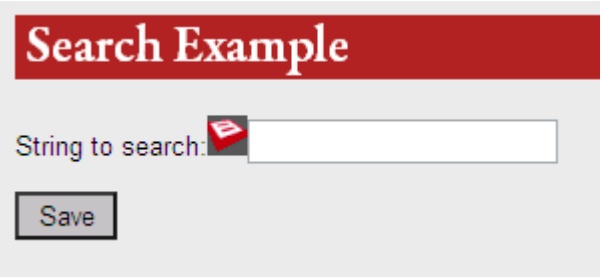
Přesto existují způsoby jak XForms v Internet Exploreru (IE) zobrazit a přestože není žádný oficiální plugin, existují řešení třetí strany, jež jsou pro IE určena.

6.1.1.1 FormsPlayer

První způsob, jak přinutit IE renderovat XForms je použít plugin třetí strany, Forms Player [http://www.formsplayer.com/], jehož instalace sice není nijak náročná (používá on-line instalátor na webu), ale je potřeba zvolit správnou verzi, protože někdy instalace „zamrzá“. S IE verze 8 funguje korektně dle mého testování verze 1.6.1030-dev.

Je-li plugin do IE úspěšně nainstalován, umožní zobrazit XForms, ale bohužel selhává na limitách IE a nedokáže otevřít soubory s MIME application/xhtml+xml. Pro zobrazení XForms je nutné je ukládat s MIME typem text/html a příslušnou koncovkou.

Řešení pomocí FormPlayeru se mi kvůli výše uvedenému nedostatku zdá velmi proprietální a dočasné a tudíž nevhodné. Je totiž nutné posílat všechny formuláře s „nekorektním“ MIME typem a navíc je ještě třeba vložit javaskriptový kód jenž uvede plugin do provozu.



Obrázek 30: Zobrazení vyhledávacího formuláře v IE pomocí Forms Player pluginu

6.1.1.2 MozzIE

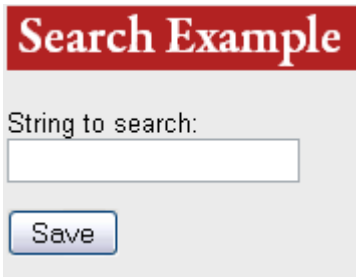
Druhý způsob je použití MozzIE (http://sourceforge.net/projects/mozzie/), což je plugin pro IE, jež umožní zobrazit XHTML a XForms pomocí Mozilla Gecko renderovacího enginu. Instalace je velmi rychlá, následně je pouze nutné ověřit, zda je doplněk v IE povolen.

V tomto případě můžeme nechat korektní MIME typ a bez použití dalších skriptů jsou formuláře renderovány i v IE.

Obrázek 31: Zobrazení vyhledávácího formuláře v IE pomocí MozzIE pluginu

6.1.2 XForms a Mozilla Firefox

Z nejpoužívanějších prohlížečů je na tom s podporou XForms nejlépe Mozilla Firefox. Sice v základní instalaci v současné době ani nejnovější verze (3.0.10) XForms přímo nepodporuje, ale lze do prohlížeče velmi jednoduše stáhnout oficiální doplněk Mozilla XForms 0.8.6ff3. Veškeré příklady v této práci (není-li uvedeno jinak) jsou testovány právě v prohlížeči Firefox za použití zmíněného pluginu.



Obrázek 32: Zobrazení vyhledávácího formuláře v Mozilla Firefox pomocí XForms extension (0.8.6ff3)

6.1.3 XForms a Opera

Opera překvapivě nemá žádný oficiální doplněk na renderování XForms a dokonce neexistuje ani žádný plugin třetí strany určený Opeře.

Přímý způsob zobrazení XForms tedy možný není, lze použít jedině některou z technik popsaných v kapitole 6.3.

6.1.4 X-smiles

Pro zobrazení XForms je také možno použít samostatný prohlížeč X-Smiles [www.xsmiles.org].

Obrázek 33: Zobrazení vyhledávácího formuláře v X-Smiles

6.2 XForms v mobilních zařízeních

Protože jazyk HTML je pro mobilní internetové prohlížeče a mobilní přenos dat pracujícím na protokolu WAP příliš obsáhlý, vznikl pro mobilní použití speciální jazyk WML (Wireless Markup Language). Webdesigneři se tak ale dostali do situace, kdy je nutné podávat různý výstup aplikace v závislosti na tom, zda návštěvník přistupuje z počítače či mobilního telefonu.

S příchodem podpory XForms a XHTML mobilními prohlížeči by tato komplikace měla úplně zaniknout, protože XForms kód formulářů bude na jakémkoli prohlížeči (který jej bude podporovat) interpretován stejně, ať už půjde o desktopový či mobilní prohlížeč.

6.2.1 Výhody použití XForms oproti WML

V rámci srovnání původních technik vytváření internetových aplikací pro mobilní telefony za použití WML a nové technologie XForms jsem se zamyslel nad výše uvedeným příkladem.

Kromě toho, že aplikace může předkládat stejný zdrojový kód jak desktopovému tak mobilnímu prohlížeči, je jasné, že XForms přinášejí na mobilní zařízení všechny své výhody, jež jsou dostupné v klasických prohlížečích s podporou XForms.

Právě XForms element Repeat, jež umožňuje opakované vkládání (v tomto případě informací o produktu), bez nutnosti okamžitého odeslání formuláře, nemá v jazyce WML vůbec žádný ekvivalent. Totožné funkcionality tak v jazyce WML v podstatě nelze dosáhnout. Vždy by bylo nutné odesílat dotaz na server a za cenu vyššího datového toku řešit přidání dalších formulářových polí serverovým skriptem.

Nová funkčnost jako například okamžitá validace formulářových polí na straně klienta či výpočet ze zadaných hodnot vede ke snížení datového toku při komunikaci na cestě klient-server, což je pro mobilní aplikace klíčové.

6.2.2 XForms a Opera Mini

Mobilní prohlížeč Opera v současné době stále nemá podporu XForms a minimálně do doby, než začne plně podporovat XHTML 2.0 ji mít nebude. To znamená, že jediné řešení, jak XForms interpretovat v mobilní Opeře Mini, je transformovat je na HTML + JavaScript (viz. 6.3.1 a 6.3.2, případně pomocí XML+XSLT) a jelikož podpora JavaScriptu v mobilních prohlížečích je značně omezená, nebude ani funkcionalita formulářů stoprocentní. Nezbývá tedy počkat, než přijde oficiální podpora.

6.2.3 XForms a Fennec

Přestože s podporou XForms je na tom nejlépe Mozilla Firefox, její mobilní bráška Fennec ji nezdědil. V mobilním prohlížeči tedy nelze použít oficiální plugin, jež používá Firefox. Dočasné řešení opět leží na transformaci XForms do HTML, což je značně neelegantní a omezující.

6.2.4 Mobilní prohlížeče s podporou XForms

Existuje však několik alternativních mobilních prohlížečů firem, jež se zabývají mobilními aplikacemi a jež už mají podporu XForms zapracovanou. Pro demonstraci zobrazení XForms na mobilním zařízení jsem si vybral programový balík společnosti DataMovil [http://www.datamovil.info/]. Obsahuje emulátor mobilního telefonu pracující na Java platformě, který zahrnuje prohlížeč XForms.

Zdrojový kód příkladu na XForms Repeat element naleznete v kapitole 5.2.3

Obrázek 34: XForms Repeat na mobilním prohlížeči DataMovil

Další společností řešící podporu XForms formulářů je například PicoForms (http://picoforms.com/). Demo jejich mobilní aplikace je dostupné zde: http://mawama.com/sbdemo.jsp?path=demo.

6.3 Univerzální řešení zobrazení XForms

Prostudoval jsem mnoho možností jak docílit zobrazení formulářů XForms v různých prohlížečích (viz. text výše) a proto jsem chtěl také najít univerzální řešení, jež by umožnilo používat XForms již v současné době. Podmínkou tedy je naservírovat prohlížeči, pro něj v současné době srozumitelný HTML kód (popř. včetně JavaScript + Ajax).

Jedna z možných cest je použít serverové technologie, jež transformují XForms kód už na straně. Zástupcem tohoto řešení je např. společnost Chiba.

Další možností je transformace až na straně klienta. Ta spoléhá čistě na JavaScriptový kód. Zástupcem je technologie FormFaces™ (více v kapitole 6.3.2).

Poslední možností by bylo vytvořit XSLT transformační styly, které by se připojovaly ke kódu XForms a každý formulářový element by transformovaly na jeho HTML ekvivalent. Takováto funkcionalita by mohla být zajišťována přímo frameworkem, jež by testoval, zda prohlížeč má či nemá podporu XForms a v závislosti na tom by odeslal XSLT. Velkým problémem tohoto řešení je absence jakékoliv možnosti validovat data již na klientovi a v podstatě se tak ztrácí všechny nové přednosti XForms. Bylo by tedy nutné následně validaci řešit na straně serveru. Řešení způsobem XML+XSLT by tedy bylo velmi pracné a v podstatě by nepřinášelo žádné výhody, kvůli kterým XForms vznikly. Bylo by třeba odvést ještě mnoho práce navíc s přípravou transformačních stylů a vzhledem k tomu, že podpora XForms v mainstreamových prohlížečích je na spadnutí, je možné, že by se tato práce stala během několika měsíců naprosto zbytečnou.

6.3.1 Transformace na straně serveru – Chiba

Chiba je servlet filtr, který umožnuje pracovat s XForms bez podpory prohlížeče. XFormsFilter (psáno v Javě) vstupuje do response serveru a XForms+XHTML data transformuje na XHTML a JavaScript, jež odesílá klientovi. Je to open-source řešení, nevýhodou však je, že předpokládá běžící Javu na serveru a tudíž zvýšenou zátěž. Více na http://chiba.sourceforge.net/.

6.3.2 Transformace na straně klienta – FormFaces™

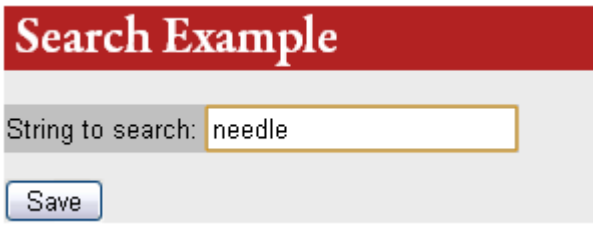
Jde o čistě JavaScriptové řešení, které používá AJAX. XForms+HTML mohou být poslány přímo prohlížeči kde JavaScript přetransformuje XForms formulářové elementy na HTML elementy. Nevyžaduje žádné řešení na straně serveru ani žádný plugin do prohlížeče. Do stránky s XForms prohlížečem stačí vložit následující JavaScript:

```
1.   <script type="text/javascript" src="formfaces.js"></script>
```

Tím se zajistí načtení souboru s JavaScriptovým kódem.

FormFaces™ slibuje kompatibilitu s následujícími prohlížeči: Internet Explorer, Netscape, Mozilla FireFox, Opera, Konquerer, Safari a NetFront. Ovšem během mého testování jsem narazil na několik chyb zejména u složitějších formulářů. Například Opera renderovala formuláře vizuálně naprosto v odlišném stylování než ostatní prohlížeče a do formulářových polí nešel zapisovat text. Také jsem zaznamenal problém s použitím XForms elementu *Repeat* či *Switch*. FormFaces™ mohou být velmi slibným nástrojem ale jejich odladění by bylo určitě časově náročnější.

Ke stažení je produkt jak v BSD tak v komerční licenci na oficiálním webu [http://www.formfaces.com/].



Obrázek 35: XForms lze díky FormFaces zobrazit například i v Google Chrome

ZÁVĚR

Hlavní pozornost této práce jsem zaměřil na technologii XForms, která byť je v povědomí webdesignérů již několik let, je pro většinu stále nová a velmi zřídka používaná. XForms jsou kvalitním produktem skupiny W3C (verze 1.0 je již standardizována) a jejich úkolem je postupně nahradit původní HTML formuláře. Vynikající užitnost a celkově vysoké ambice XForms jako technologie webových formulářů budoucnosti dokazují teoretické předpoklady i jejich praktické používání. Přestože mají složitější konstrukci jednoduchých formulářů, podstatně usnadňují tvorbu těch složitějších, kde je v současnosti nutné použít skriptování.

Tato práce však není pouze o teoretickém představení XForms technologie, ale také o její praktické implementaci do PHP frameworku Qcodo. Qcodo je open-source vývojový framework, jež díky své komunitě slibuje další rozvoj a samotní členové komunity ocenili možný budoucí přínos nových knihoven pro práci s XForms. Nové knihovny, jež jsem vytvořil, umožní vývojáři definovat objekty formuláře podobně, jak tomu byl dosud ve frameworku Qcodo zvyklý, ovšem místo HTML výstupu generují kód XForms. Pro rychlý vývoj aplikací jsem změnil i šablony ORM generátoru kódu, pomocí nějž lze přímo z modelu databáze nechat automaticky vygenerovat třídy pro práci s databázovými tabulkami a dokonce i jednoduché návrhové administrační prostředí s formuláři XForms.

Masovějšímu rozšíření XForms ještě v současnosti stále chybí podpora internetových prohlížečů. Jelikož hlavní přínos XForms tkví v jednoduchosti, univerzálnosti (stejný kód pro desktopový i mobilní prohlížeč) a v omezení komunikace klient-server, jakékoliv snahy o transformaci XForms do původních technologií HTML+JavaScript se mi jeví jako dočasné a neefektivní řešení. Veškerá genialita technologie webových formulářů budoucnosti tedy padá se základním problémem. Pro běžného uživatele jsou zatím nedostupné a pokud ne, tak pouze za cenu zbytečně komplikovaných řešení, jež stejně ve finále používají techniku původních HTML formulářů.

XHTML 2.0 se dle konsorcia W3C nejspíše brzy stane doporučením a tím okamžikem by v jazyce XHTML měly XForms definitivně nahradit stávající HTML formuláře. Je tedy pouze otázkou času, kdy se podpora rozšíří do hlavích desktopových a mobilních internetových prohlížečů a těm připraveným přinese nasazení XForms veškeré očekávané výhody.

CONCLUSION

The main attention of this work, I focused on XForms technology, which exists few years, but for most web designers is still new and very rarely used. XForms is W3C group product (version 1.0 is already standardized), and its task is to replace the original HTML form. Better user experience and overall high ambitions of XForms technology as the future of web forms show the theoretical assumptions and their practical application. Although the design of simple form is little bit more difficult, making complex forms in which is currently necessary to use scripting is easier.

This work is not only a theoretical presentation of XForms technology, but also its practical implementation in PHP Framework Qcodo. Qcodo is open-source development framework, with active community, which promises further development, and community members appreciate the potential future benefits of new libraries for handling with XForms. The new libraries that I created will allow developers to define form element objects in the same way as they has been used to, but instead of HTML output it generates XForms code. For rapid application development, I changed the templates from ORM code generator, using which you can directly from the model database automatically generate a class for working with database tables and even with a simple drafts (GUI administration) which uses XForms.

Greater use of XForms is currently still lacking support for web browsers. The main contribution of XForms lies in simplicity, universality (the same code for desktop and mobile browser) and less of client-server communication and therefore effort to transform XForms to indigenous technologies HTML + JavaScript will bring only temporary and ineffective solution. All the best point of XForms web technology of the future falls because of the fact, that in this situation are XForms unavailable for the current user.

XHTML 2.0 according to W3C consortium probably soon become a recommendation and at the moment in XHTML language XForms should definitely replace the existing HTML forms. It is therefore only a matter of time before the support extends to the mainstream desktop and mobile web browsers and ready developers will start to use the XForms technology and will profit from it.

SEZNAM POUŽITÉ LITERATURY

[1] *The Forms Working Group* [online]. 1999-2007 , 2009/03/05 12:53:43 [cit. 2009-04-24]. Text v angličtině. Dostupný z WWW: <http://www.w3.org/MarkUp/Forms/#documents>.

[2] YOUNG, Michael J. *XML krok za krokem*. 2. aktualiz. vyd. Brno : Computer Press, a.s., 2006. 471 s.

[3] *Extensible Markup Language* [online]. 2009 , 18. 4. 2009 15:14. [cit. 2009-04-24]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Extensible_Markup_Language>.

[4] *XSLT* [online]. 2009 [cit. 2009-04-30]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/XSLT>.

[5] Jiří Kosek. „Základy jazyka WML“. 74-76 s. In: *Mobil*. 10/2000. ISSN 1211-6343. Dostupné na WWW: <http://www.kosek.cz/clanky/wapkurz/wml-zaklady.html>.

[6] *XML Path Language (XPath) : Version 1.0* [online]. 1999 [cit. 2009-05-19]. Dostupný z WWW: <http://www.w3.org/TR/xpath#section-Introduction>.

[7] ŽÁK, Miroslav. *XML : (začínáme programovat)*. Praha : Grada Publishing, a.s., 2003. 214 s. ISBN 80-247-0565-6.

[8] *XForms 1.0 Frequently Asked Questions : W3C Forms Working Group* [online]. [2002] [cit. 2009-04-24]. Text v angličtině. Dostupný z WWW: <http://www.w3.org/MarkUp/Forms/2003/xforms-faq.html>.

[9] EISENBERG, J. David. *Using XForms with Mozilla*. [s.l.] : O'Reilly Media, Inc. , 2007. 46 s.

[10] ROZEHNAL, Marek, DAňEK, Petr. *Velký test PHP frameworků* [online]. 2008 [cit. 2009-05-18]. Dostupný z WWW: <http://www.root.cz/clanky/velky-test-php-frameworku-2008/>.

[11] *Qcodo - PHP Development Framework : Code Less. Do More.* [online]. c2005-2009 [cit. 2009-05-18]. Dostupný z WWW: <http://www.qcodo.com/>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTML Hypertext Markup Language - značkovací jazyk určen pro tvorbu online dokumentů.

MIME Multipurpose Internet Mail Extensions - internetový standard definující typ souboru (původně využití v elektronické poště)

MVC Model-view-controller - softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.

ORM Object-relational mapping - v počítačové technice chápáno jako programovací technika pro konverzi dat mezi nekompatibilními systémy v relačních databázích a objektové orientovaném programování.

PHPDoc Kompletní řešení dokumentace PHP kódu. Používá standardní formát dokumentace a automatické generování.

W3C World Wide Web Consorcium - je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy.

WML Wireless Markup Language - značkovací jazyk založený na jazyce XML určen pro tvorbu online dokumentů pro mobilní zařízení.

CRUD Create, read, update and delete - základní funkce úložiště dat (databáze)

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: SQL KÓD MODELU DATABÁZE

```
1.  CREATE TABLE user_account (
2.    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
3.    nick VARCHAR(255) NOT NULL,
4.    mail VARCHAR(255) NOT NULL,
5.    PRIMARY KEY(id),
6.    UNIQUE INDEX user_account_index1247(nick, mail)
7.  )
8.  TYPE=InnoDB;
9.
10. CREATE TABLE category (
11.   id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
12.   name VARCHAR(255) NOT NULL,
13.   description VARCHAR(255) NULL,
14.   PRIMARY KEY(id)
15. )
16. TYPE=InnoDB;
17.
18. CREATE TABLE user_personal (
19.   id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
20.   user_account_id INTEGER UNSIGNED NOT NULL,
21.   name VARCHAR(255) NOT NULL,
22.   street VARCHAR(255) NULL,
23.   postal_code INTEGER(5) UNSIGNED NOT NULL,
24.   city VARCHAR(255) NOT NULL,
25.   PRIMARY KEY(id),
26.   INDEX user_personal_FKIndex1(user_account_id),
27.   FOREIGN KEY(user_account_id)
28.     REFERENCES user_account(id)
29.       ON DELETE NO ACTION
30.       ON UPDATE NO ACTION
```

```
)
TYPE=InnoDB;

CREATE TABLE product (
  idproduct INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  category_id INTEGER UNSIGNED NOT NULL,
  name VARCHAR(255) NULL,
  description TEXT NULL,
  price FLOAT NULL,
  PRIMARY KEY(idproduct),
  INDEX product_FKIndex1(category_id),
  FOREIGN KEY(category_id)
    REFERENCES category(id)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
TYPE=InnoDB;

CREATE TABLE product_order (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  product_idproduct INTEGER UNSIGNED NOT NULL,
  user_account_id INTEGER UNSIGNED NOT NULL,
  date DATE NULL,
  PRIMARY KEY(id),
  INDEX shopping_cart_FKIndex1(user_account_id),
  INDEX shopping_cart_FKIndex2(product_idproduct),
  FOREIGN KEY(user_account_id)
    REFERENCES user_account(id)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(product_idproduct)
```

```
62.       REFERENCES product(idproduct)
63.         ON DELETE NO ACTION
64.         ON UPDATE NO ACTION
65.   )
66.   TYPE=InnoDB;
67.
```